Machine Learning Applied To Build 'Nowcasting' Models to Predict Irish Rainfall

Abhishek Vilas Kulkarni

The thesis is submitted to University College Dublin
in part fulfilment of the requirements for the degree of
MSc Data and Computational Science

School of Mathematics and Statistics
University College Dublin

Supervisor: Dr. Xuefeng Cui
Dr. Ray McGrath

August 30, 2019

Abstract

Rain forecasting plays a significant part in our daily lives. Just as long term forecasting, the short term forecasting (Nowcasting) is important as well. While planning for activities, such as outdoor sports, daily commute or even tours, it becomes imperative to check the rain first. Nowcasting is also of interest in airports since extreme conditions, particularly strong winds or storms, are not desired while flight take-off/landings. In this study, the rainfall for the next 12 hours is forecasted based on the statistical and the machine learning techniques, rather than the traditional Numerical Weather Prediction (NWP) techniques that rely on computationally intensive models / machines. Though rainfall is forecasted to the next 12 hours in this study, the focus is on very short term rain forecasting for up to 3-6 hours. The goal is to test whether the statistical and the machine learning techniques have any skill (over "persistence") in very short-range forecasting of the the occurrence (or non-occurrence) and intensity of hourly rainfall. The study has focused on rainfall (rather than wind speeds, temperature or other atmospheric parameters) as it is a very "noisy" parameter that heavily challenges machine learning techniques. The data is gathered from the Dublin Airport weather station. The historical hourly Rainfall data with several parameters such as, Pressure, Temperature, Specific Humidity and Wind Speed, is modelled as times series. The most significant parameters that determine future rainfall are determined. The stationary time series rainfall data are fitted with the Vector AutoRegression (VAR), the AutoRegressive Integrated Moving Average (ARIMA) and the Neural Network models in order to determine which method yields the most accurate forecasts. The persistence model, where the last observed rainfall is forecasted for the next twelve hours, is considered as the baseline and the other models are compared to this baseline model to know the skills of these models. While the persistence model seems very basic, it has considerably good predicting power, since if it is dry weather now, it is likely to be dry in the next few hours. The accuracy metrics Root Mean Squared Errors (RMSE) and Mean Absolute Percentage Errors (MAPE) are used as the performance measures. In addition, skill scores, which are based on Mean Squared Errors(MSE), are also used to determine how the given model performs compared to the baseline model. Several case studies are conducted to understand how the different models differ in forecasting rainfall. While the hourly case studies show different cases of varying rainfall throughout the day and the respective forecasts of the models, the monthly case studies compares the different hourly forecasts of the models. On examination of RMSE, MAPE and skill score values, and the hourly and monthly case studies, it is found that the VAR model is the most accurate model, closely followed by the ARIMA model. Though Neural Network model took the last place in terms of the accuracy, it is better for accurately forecasting imminent peaks in rainfall.

Acknowledgments

I would like to express my sincere gratitude to my supervisors Dr. Xuefeng Cui and Dr. Ray McGrath for providing their invaluable guidance, comments and suggestions throughout the course of the project. I heartily appreciate the wisdom and experience that they shared during our weekly meetings.

I would also like to extend my gratitude to my Module Coordinator, Dr. Barry Wardell and to my Project Coordinator, Dr. James Herterich for their guidance throughout the semester.

I would like to thank all my professors for teaching me the essential concepts without which it would not have been possible to work on this project.

Finally, I would like to thank all the people, especially Gourishankar Bawade, who encouraged and supported me to complete this research project successfully.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Predicting the climate has been practiced since the beginning of time with more or less accuracy. Although meteorologists' observations aren't always precise, having little knowledge about what may happen is better than none. Without meteorologists in weather stations preparing weather forecasts for us, we wouldn't know what to expect from the weather. Just as the long term weather forecasting, the short term forecasting (Nowcasting) is crucial as well.

1.2 Motivation and Goal

Weather forecasting plays an important part in our daily lives since knowing the weather conditions enables us to prepare for the type of weather we will be facing. Be it morning, afternoon, or night, rainfall impacts our daily lives in several ways. While planning for activities, such as outdoor sports, daily commute or even tours, it becomes imperative to check the rain first. In considerably wet regions like Ireland, forecasts are very helpful in guiding us regarding the atmospheric conditions. In the airports, the pilots in flights need to know about the local weather prior to take-off or landing as extreme conditions, particularly strong winds or storms, are not desired. A short term forecast could help them to a greater extent.

Traditionally, Numerical Weather Prediction (NWP) techniques[1] are employed to forecast the weather. These methods require considerable amount of time to run and give the forecasts. Since forecasts in short term are desired, these techniques may be out-of-date by the time the forecasts are generated. On the other hand, the Machine Learning and Statistical Models are generally much faster to train and the forecasts are quite reliable.

Therefore, the primary aim of this study is to accurately forecast the rainfall on a short

term period by using the Statistical and Machine Learning techniques. Though rainfall is forecasted to the next 12 hours in this study, the focus is on very short term rain forecasting for upto 3-6 hours. Rather than the traditional Numerical Weather Prediction (NWP) techniques that rely on computationally intensive models / machines, Machine Learning approaches are employed here. The goal is to see whether machine learning techniques can be usefully employed in this field.

1.3 Data

The data is gathered from the Dublin Airport weather station. From this weather station, the hourly surface level weather parameters, Rainfall (in millimeters), Pressure (in hPa), Dew Point (in degrees Celsius), Air Temperature (in degrees Celsius), Wind Direction (in degrees - 0 to 360) and Wind Speed (in knots) are gathered from Met Éireann, the Irish National Meteorological Service. Additionally, the weather parameters Temperature (in degrees Celsius), Wind Direction (in degrees - 0 to 360), Wind Speed (in knots) and Specific Humidity from five different pressure levels above the surface are collected from the ERA-Interim Reanalysis Project. The five pressure levels include 925, 850, 700, 500 and 200 hPa which are at the approximate heights of 750, 1500, 3000, 5500 and 12000 meters above the surface respectively. The data are hourly measurements of the mentioned weather parameters from 1981 to 2015.

Chapter 2

Theory and Methods

2.1 Theory

Several studies have shown that the ARIMA Model is well suited for analysing rainfall time series. These studies[2][3] show that rainfall time series in Thailand and Gaza respectively fitted with the ARIMA model, which is built on the basis of exponential smoothing, gave accurate forecast results. Further, the study[4] conducted at Beijing and Haikou also showed good results when rainfall data are fitted with the ARIMA model. Hence, the ARIMA model is employed in this study.

Some research studies have also shown that the rainfall data modelled with the artificial neural networks gave accurate forecasts. For instance, this study[5] conducted at Nilgiris, India showed that the feed forward neural network predicted the rainfall precisely. Similarly, this study [6] also verified good accuracy of ANN in India. The other study[7] taken up in Indonesia proves VAR model to be highly accurate.

All these studies considered yearly/monthly/daily rainfall and the respective model was fitted correspondingly. This study involves in the hourly rainfall and the interest lies in very short term forecasting.

2.2 Preliminary Analysis

Missing Data

Out of 306792 observations, only 507 observations are missing, which accounts for less than 1 percent of the entire data. Since the data are modelled as time series data, it is imperative to record the data at each unit time. So, data imputation (replacing missing data with substituted values) is done using linear interpolation method[8] on the respective variables and the time series data is preserved. Contrary to the traditional data imputation, time series data imputation needs to fill up the gaps in time dependent variables. Hence, the interpolation method is employed in this study.

Data Analysis

The Figure 2.1 shows how the rainfall occurs hourly from 1981 to 2015 at Dublin Airport weather station.

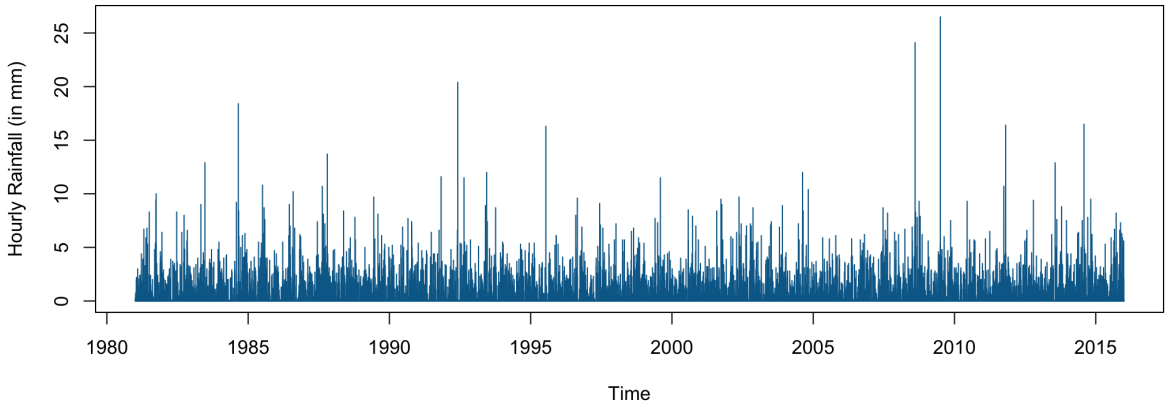

Figure 2.1: Time Series Plot showing Rainfall from 1981 to 2015

Evidently, lot of observations are below 5 mm per hour and very few observations are above 10 mm per hour. Also, large number of observations have zero hourly rain. This is witnessed in the Figure 2.2, which depicts the density of Log-rainfall. Lot of observations have 0 rainfall and very few observations have heavy rainfall. So, it becomes quite challenging to model such skewed data.

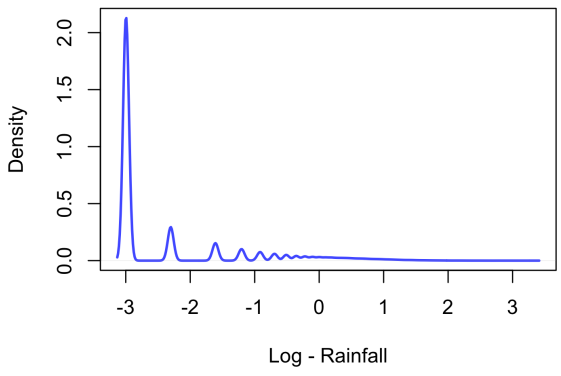

Figure 2.2: Rainfall Density Plot

The Rainfall is decomposed into its constituent components: trend, seasonality and random. The Figure 2.3 shows the decomposed time series data[9], modelled as

$$Y_t = T_t \times S_t \times R_t$$

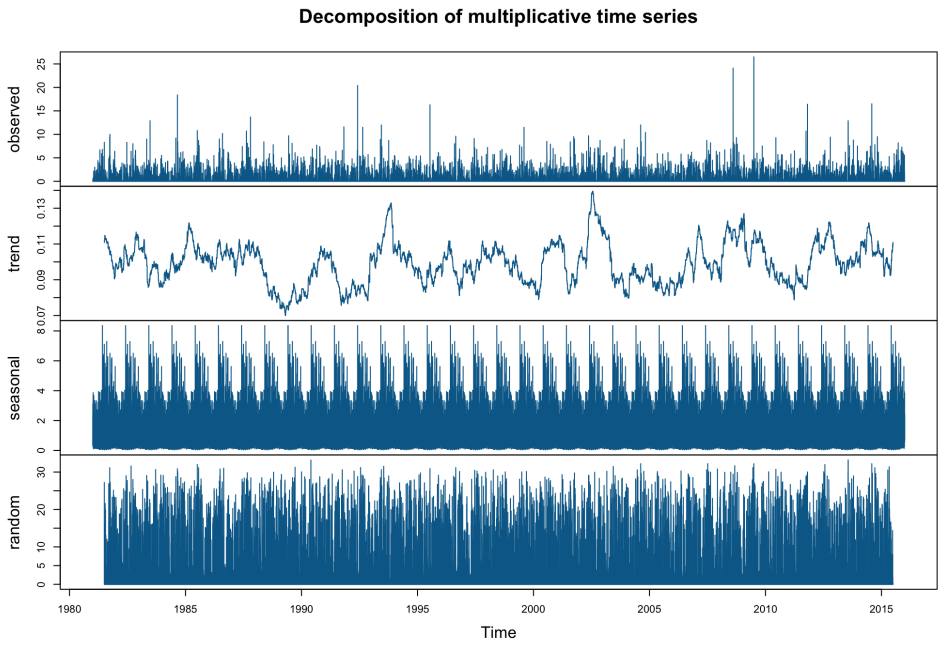

Figure 2.3: Decomposition of Rainfall Time Series into components: Trend, Seasonal and Random components

where Y_t is the rainfall data, T_t is the trend-cycle component, S_t is the seasonal component and R_t is the remainder random component at the time t.

As witnessed in the figure, trend depicts how the rainfall varied through the years. The seasonal component shows repeating short-term cycle in the rainfall series. Then, the random component shows the random variation in rainfall. Evidently, range of random component is very high.

Statistics	**Rainfall**	**Pressure**	**Dew**	**Air Temp**	**Wind Dir**	**Wind Speed**
Minimum	0	945.54	-10.25	-11.5	0	0
First Quartile	0	1005.9	3.67	6.2	140	6
Median	0	1014.6	7.16	9.8	230	10
Mean	0.099	1013.434	6.963	9.732	203.749	10.373
Third Quartile	0.05	1022.1	10.31	13.3	260	14
Maximum	26.5	1047.2	20.23	28.5	360	45
Std Deviation	0.420	12.404	4.450	4.914	85.762	5.770

Table 2.1: Summary Statistics of Surface level Parameters

The summary statistics for all surface level parameters are calculated and recorded in Table 2.1. The median of rainfall is 0, indicating that the weather is dry for most hours. The wind direction represents the direction of the origin of wind, represented as an angle between 0° and 360°. Air temperature and Dew point are expressed in degree Celsius and Pressure is expressed in hPa. The summary statistics of parameters recorded at higher altitudes (at 5 different pressure levels) are tabulated in Table 2.2. As compared

to the surface level parameters, the upper level temperatures and wind speeds show much variation due to the high altitude. The Specific Humidity indicates the moisture content present in the atmosphere.

Statistics	Temperature	Wind Direction	Wind Speed	Specific Humidity
Minimum	-74.10	0.0	0.00	0.000000
First Quartile	-24.61	190.9	16.20	0.000232
Median	-5.47	247.9	26.60	0.001565
Mean	-14.77	227.6	30.84	0.002366
Third Quartile	2.73	288.8	40.50	0.004072
Maximum	23.57	360.0	171.00	0.014302
Std Deviation	23.098	87.474	20.087	0.002379

Table 2.2: Summary Statistics of Parameters recorded at higher altitudes

A Stationary time series is one whose properties (mean and variance) do not depend on the time at which the series is observed. In time series analysis, stationarity plays a prominent role. This is because, any model built with non-stationary data will vary in accuracy at different points of time. To check if the rainfall data is stationary, The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test[10] is performed. The results are shown in Figure 2.4. The null hypothesis for this test is that the data is stationary and the alternate hypothesis is that the data is not stationary. As the p-value is higher than 0.05, the null hypothesis is not rejected at the usual 5% level of significance. Hence, the rainfall data is stationary over the selected time period.

```
        KPSS Test for Level Stationarity

data:  Rainfall
KPSS Level = 0.23404, Truncation lag parameter = 29, p-value = 0.1
```

Figure 2.4: Results of KPSS Test showing that the rainfall data is stationary

```
        Augmented Dickey-Fuller Test

data:  Rainfall
Dickey-Fuller = -59.03, Lag order = 67, p-value = 0.01
alternative hypothesis: stationary
```

Figure 2.5: Results of ADF Test showing that the rainfall data has no unit root

A unit root is a stochastic trend in a time series. In presence of a unit root, the time series shows a systematic pattern that is hard to predict. To test for absence of unit root in rainfall data, the Augmented Dickey Fuller Test (ADF)[11] is performed and the results

are recorded in Figure 2.5. The null hypothesis for this test is that there is a unit root and the alternate hypothesis is that the time series is stationary. Since the p-value is less than 0.05, the null hypothesis is rejected at 5% significance level. Therefore, rainfall data is stationary and has no unit root over the selected time period.

Feature Selection

Out of all the surface level parameters and upper level parameters, only the parameters that greatly influence future rainfall and help in predicting rain are useful. To determine the significant parameters, Time Series Linear Model[12] is constructed and the coefficients of the model are examined. The coefficients of the Time Series Linear Model constructed with Rainfall as the response variable and all other variables as the predictors are tabulated in Table 2.3. It follows from the table that the surface level parameters (Pressure, Temperature, Dew Point, Wind speed) and the Specific Humidity of the upper level are significant at 5% level of significance. Hence, the surface level parameters and the Specific Humidity of the upper levels are taken to build the models as these parameters lead to lower AIC (Akaike Information Criterion) values when the models are constructed and the predictions are comparatively accurate.

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)             6.759e+00  5.442e-01  12.420  < 2e-16 ***
Pressure               -6.047e-03  5.317e-04 -11.373  < 2e-16 ***
Air_Temp               -3.718e-02  2.524e-03 -14.732  < 2e-16 ***
Dew_Point               3.001e-02  4.195e-03   7.152 9.22e-13 ***
Wind_direction         -1.811e-04  6.375e-05  -2.841  0.00451 **
Wind_speed              9.347e-03  1.458e-03   6.410 1.53e-10 ***
X200.Temperature        2.266e-03  9.385e-04   2.414  0.01580 *
X200.Wind_direction     1.151e-04  7.026e-05   1.638  0.10135
X200.Wind_speed        -1.653e-04  2.965e-04  -0.558  0.57716
X200.Specific_Humidity -3.511e+03  7.882e+02  -4.454 8.52e-06 ***
X500.Temperature        8.543e-03  2.621e-03   3.259  0.00112 **
X500.Wind_direction    -9.847e-05  8.072e-05  -1.220  0.22255
X500.Wind_speed         3.545e-04  5.471e-04   0.648  0.51698
X500.Specific_Humidity  5.028e+01  1.159e+01   4.336 1.47e-05 ***
X700.Temperature        6.563e-03  3.273e-03   2.005  0.04497 *
X700.Wind_direction     3.985e-06  8.290e-05   0.048  0.96166
X700.Wind_speed         7.068e-04  8.691e-04   0.813  0.41610
X700.Specific_Humidity  5.102e+01  5.420e+00   9.413  < 2e-16 ***
X850.Temperature       -8.218e-03  3.153e-03  -2.607  0.00916 **
X850.Wind_direction    -2.510e-04  8.752e-05  -2.868  0.00414 **
X850.Wind_speed        -2.366e-03  1.175e-03  -2.014  0.04404 *
X850.Specific_Humidity  1.708e+01  5.451e+00   3.133  0.00174 **
X925.Temperature       -2.285e-03  3.644e-03  -0.627  0.53059
X925.Wind_direction    -1.728e-04  8.310e-05  -2.079  0.03766 *
X925.Wind_speed         7.109e-05  1.180e-03   0.060  0.95195
X925.Specific_Humidity -1.897e+01  7.641e+00  -2.483  0.01304 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Table 2.3: Coefficients of Time Series Linear Model

2.3 Model Fitting

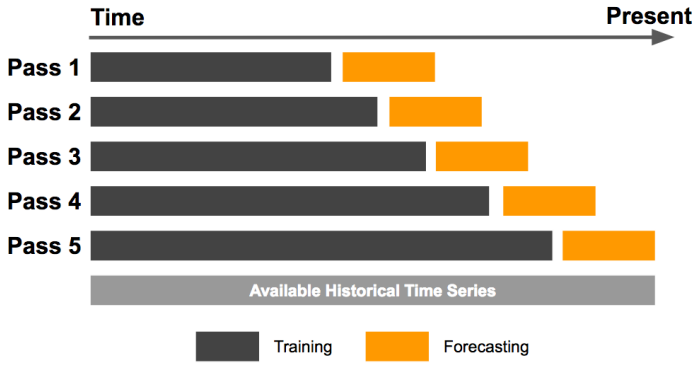

Figure 2.6: Training models using Expanding Window

For each method, the data from 1981 to 2009 is taken as training data and the data from 2010 is taken as test data. Since the size of test data is sufficiently large (8760 rows), robust results are generated. The forecasting is performed for 12 hours by increasing the training window by 1 hour[13]. Such model is referred to as expanding window[14]. So, in the first iteration, first 12 hours (1st January, 2010) are forecasted[15][16]. In the second iteration, the first hour is included in the training window and the hours 2-13 are forecasted. This process is continued for the whole year. This method is employed since it gives robust results. This expanding window method to train the models can be visualized as shown in Figure 2.6.

2.4 Persistence Model

In univariate models, only the hourly rainfall is considered. The models are trained with past hourly rainfall and the rainfall for the next 12 hours is forecasted.

Persistence Model is one of the most basic models. The last observed rainfall is forecasted for the next twelve hours. Essentially, only the last observation of rainfall is required to forecast. If Y_t is the rainfall at time t, the forecast for next 12 hours is

$$Y_{t+h} = Y_t$$

where $h = 1, 2, \ldots, 12$.

This model assumes that the atmospheric conditions will remain unchanged during the forecast time. The rationale behind this is that, if the weather is dry at the present hour, it is likely to be dry in the near future (12 hours). While this models seems considerably inaccurate, it proves to be quite precise at most times, partly since the rainfall data is zero-inflated.

Although this model performs well when the conditions are in a steady state, poor forecasts are inevitable when the weather conditions are fluctuating. In other words, there is no skill in forecasting using this model. Hence, this model is used as a baseline in this study to compare model performances.

2.5 Vector AutoRegression Model

In multivariate models, along with the past rainfall, other parameters, such as temperature and wind speed, are also utilized for fitting the data to the model. Then, the rainfall is forecasted for the next 12 hours, based on change in past rainfall and other parameters.

In the VAR framework, all variables are treated symmetrically. They are all modelled as if they all influence each other equally. A VAR model is a generalization of the univariate autoregressive model for forecasting a vector of time series. It comprises one equation per variable in the system. The right-hand side of each equation includes a constant and lags of all the variables in the system[12].

The VAR (Vector AutoRegression) model[17][18][19] is known to capture the linear inter-dependencies among the multiple time series data. Each variable has an equation explaining its evolution based on its own lagged values, the error term and the lagged values of other variables in the model. So, the past lagged values of rainfall, past lagged values of all other variables and the corresponding errors are considered to forecast the rainfall. A VAR(p) model equation for rainfall is represented as:

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t$$

where, y_t is the rainfall at time t, y_{t-i} is the i-th lag of y, c is a vector of constants, A_i is a $k \times k$ matrix and e_t is the vector of error terms. The models are estimated equation by equation using the principle of the least squares. For each equation, the parameters are estimated by minimizing the sum of squared e_t values.

2.6 ARIMA Model

ARIMA Model (AutoRegressive Integrated Moving Average) is known to capture different temporal structures in time series data[20]. The AR part of ARIMA indicates that the variable (rainfall) regressed on its own prior values. The MA part indicates that regression errors are a linear combination of error terms whose values occurred at various times in the past. The 'I' indicates that the differencing of data values may be employed if the time series data is non-stationary. These features help in prediction as accurate as possible.

An ARIMA(p,d,q) model is represented as:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

where, y_t is the time series data (differenced if non-stationary), p is the order of the autoregressive (AR) component, q is the order of the moving average (MA) component and d is the number of times the series has been differenced to obtain stationarity.

The stationarity of the time series data is tested by ADF and KPSS tests and the Auto Correlation Function (ACF) plots of the data points. The trends in the time series data and constant variance assumption can be analyzed using Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF) plots. Based on certain characteristics of the ACF graphs, successive differencing of the data series might be used until the data is concluded to be stationary. The autoregressive and moving average terms can also be determined using the ACF and PACF plots. Based on the identified model structure (p, d, q), the identifying model parameters need to be obtained and the fitted model can then be used to forecast the future values of rainfall.

2.7 Time Series Neural Network Model

A neural network can be thought of as a network of "neurons" which are organised in layers. The predictors form the input layer, and the forecasts form the outputs layer. There may also be intermediate layers containing "hidden neurons". The appropriate weights are attached to these predictors and the forecasts are obtained by a linear combination of the inputs. The weights are selected in the neural network framework using a learning algorithm that minimizes a cost function such as the MSE (Mean Squared Errors)[12].

The neural network is known to capture the non-linear trend in the time series data. Here, there are p lagged inputs of rainfall, k nodes in the hidden layer and the forecasted rainfall is the output. A feed-forward neural network[21] is fitted with lagged values of rainfall and other variables as inputs and a single hidden layer. The resulting neural network has 12 input nodes comprised of other variables and lagged values of rainfall, a hidden layer with 6 nodes and 1 output node as shown in Figure 2.7.

The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output. The inputs into hidden neuron j in Figure 2.7 are combined linearly to give

$$z_j = b_j + \sum_{i=1}^{12} w_{i,j} x_i$$

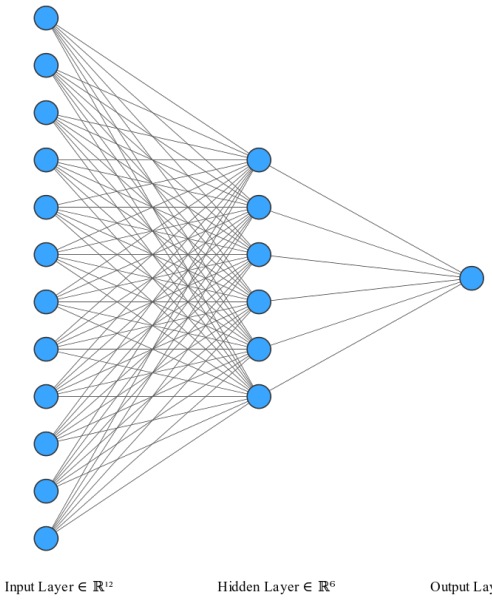

Figure 2.7: Time Series Neural Network Model with 12 input nodes, single hidden layer with 6 nodes and 1 output node

In the hidden layer, this is then modified using a nonlinear function such as a sigmoid,

$$s(z) = \frac{1}{1 + e^{-z}}$$

to give the input for the next layer. This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers. The parameters $b_1, b_2, \ldots, b_{12}$ and $w_{1,1}, \ldots, w_{12,6}$ are learned from the data.

2.8 Performance Measures

2.8.1 Accuracy Metrics

In order to compare the accuracies of the models, the root mean square error (RMSE) is calculated for each of the models. Lower this accuracy metric, better the model is at forecasting rainfall. RMSE is calculated as the square root of the average of squared differences between forecasted and actual observations[22]. This measure also tends to exaggerate large errors, which can help eliminate methods with large errors. RMSE penalizes large errors due to the squared term.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

where, y_t is the observed rainfall, $\hat{y}_t$ is the forecasted rainfall at time t and n is the total number of observations.

With the advantage of being unit free, percentage errors are very often used to compare the forecast performance of different models. MAPE (Mean Absolute Percentage Error) is a relative error measure that uses absolute values. The absolute values keep the positive and negative errors from cancelling out each other giving a fair measure. Since relative errors do not depend on the scale of the dependent variable, this measure is suitable to compare the forecast accuracy between differently scaled time series data.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} |\frac{y_t - \hat{y}_t}{y_t}|$$

where, y_t is the observed rainfall, $\hat{y}_t$ is the forecasted rainfall at time t and n is the total number of observations.

2.8.2 Persistence and Skill

The skill of a forecast model measures the superiority of the forecast over a simple baseline model. The persistence model is considered as the baseline in this study since it requires no skill to forecast. With this model as a baseline, the skill scores of the models are computed as:

$$SkillScore_{forecast} = \frac{MSE_{forecast} - MSE_{baselinemodel}}{MSE_{perfectforecast} - MSE_{baselinemodel}}$$

where, $MSE_{forecast}$ is the Mean Squared Error of the respective forecast model, $MSE_{baselinemodel}$ refers to the MSE of the persistence model and $MSE_{perfectforecast} = 0$.

Chapter 3

Results and Discussion

3.1 Hourly Case Studies

To have a deeper understanding on how the models behave, several case studies are done by comparing observed rainfall with the forecasted rainfall. As described in section 2.3, the training period is increased by 1 hour and the rainfall for the next 12 hours is forecasted using different methods. This process is iteratively executed for the year 2010. In these case studies, a 24-hour period is considered and the forecasts started at different points are observed along with the actual rainfall. This will illustrate how the models perform by iteratively increasing the training window. Four such cases are considered:

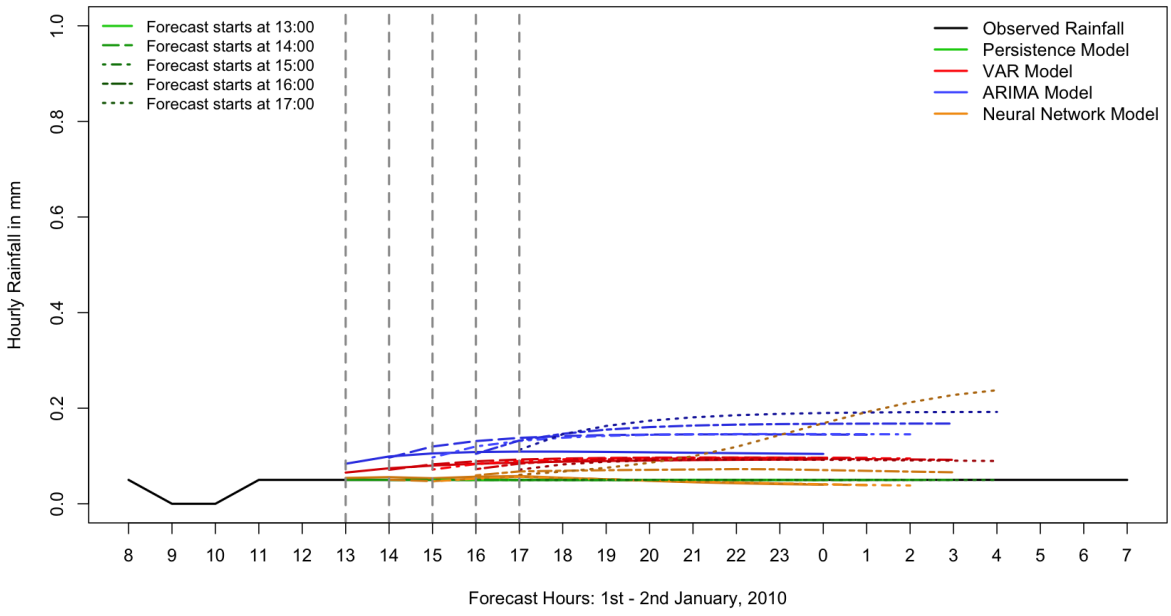

Figure 3.1: Case Study: Rain forecasts of different models when there is no rainfall throughout the period

Case 1: No Rainfall throughout the period

In this case, there is no rainfall (or very little rainfall) throughout the day. The Figure 3.1 shows this case where the rainfall from 8am on 1st January to 7am on 2nd January is observed. The forecasts are started at 5 different points of time: 13:00, 14:00, 15:00, 16:00 and 17:00 hours. The forecasts of different models are colour-coded and the forecasts started at different points have different line-types. It follows that the persistence model is accurate in this case as there is no rainfall throughout the period. The forecasts from the models predicted very little rainfall. It is also evident that the VAR model forecasts tend to converge to a point and the ARIMA model forecasts tend to be flat after a few hours. The Neural Network model forecasts are very close to the actual rainfall, but some errors are inevitable in all the models.

Case 2: Rain starts pouring after a long dry weather

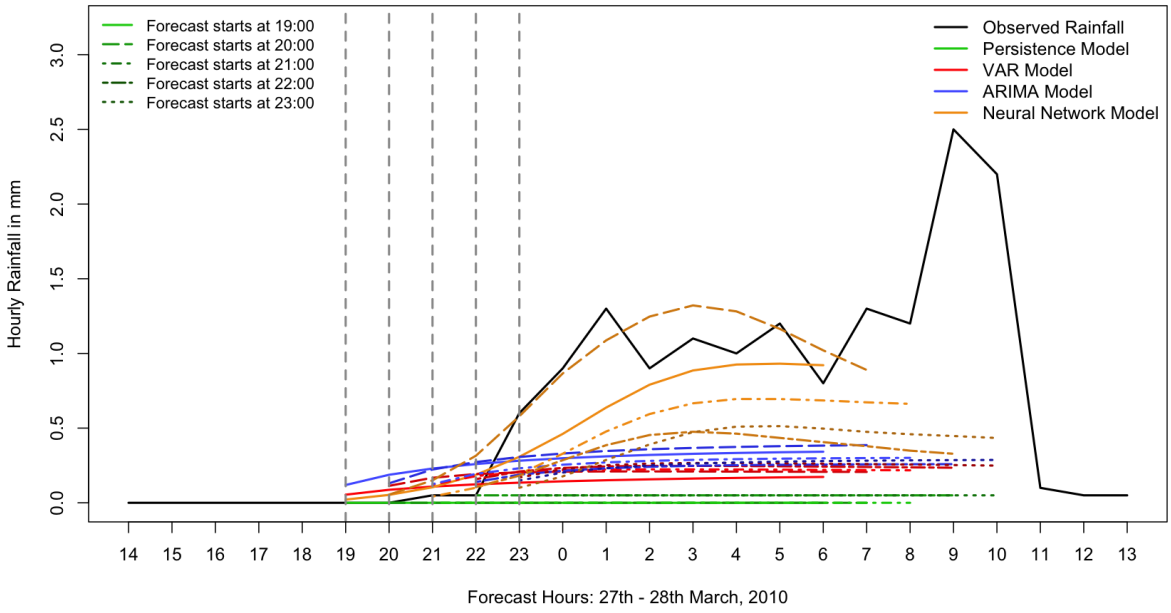

Figure 3.2: Case Study: Rain forecasts of different models when rain starts pouring after a long dry weather

In this case (27th - 28th March, 2010), rainfall picking up from zero rainfall after a long dry weather is considered to understand how the different models behave. It follows from Figure 3.2 that the persistence model performed poorly as the trend was not picked up. The VAR model and the ARIMA model picked up some non-zero rainfall, but the extent is quite small. The Neural Network model performed really well compared to other models in this case and the forecast picked up the approximate trend in the rainfall. The

range of Neural Network resembles that of the actual rainfall and this model performed the best in this case study.

Case 3: Rain stops after a peak

The case where rainfall stops after a peak rainfall (28th - 29th October, 2010) is shown in Figure 3.3. This case is considered to check whether the models pick up this trend and this is quite a usual case after a long rainfall. As before, the persistence model did not perform well as only the last hour's rainfall is used to forecast for the next 12 hours. The ARIMA model forecasted incorrectly initially. But when the observed rainfall dropped from 7am to 8am, the model correctly forecasted low rainfall. The VAR model performed consistently well, where the low trend in forecast is seen throughout the 5 starting points. The Neural Network model performed badly at 6am and 7am as high rain was forecasted. Once the rainfall dropped, the proper low forecast trend is seen beyond 8am. Overall, VAR model performed well through all the starting points.

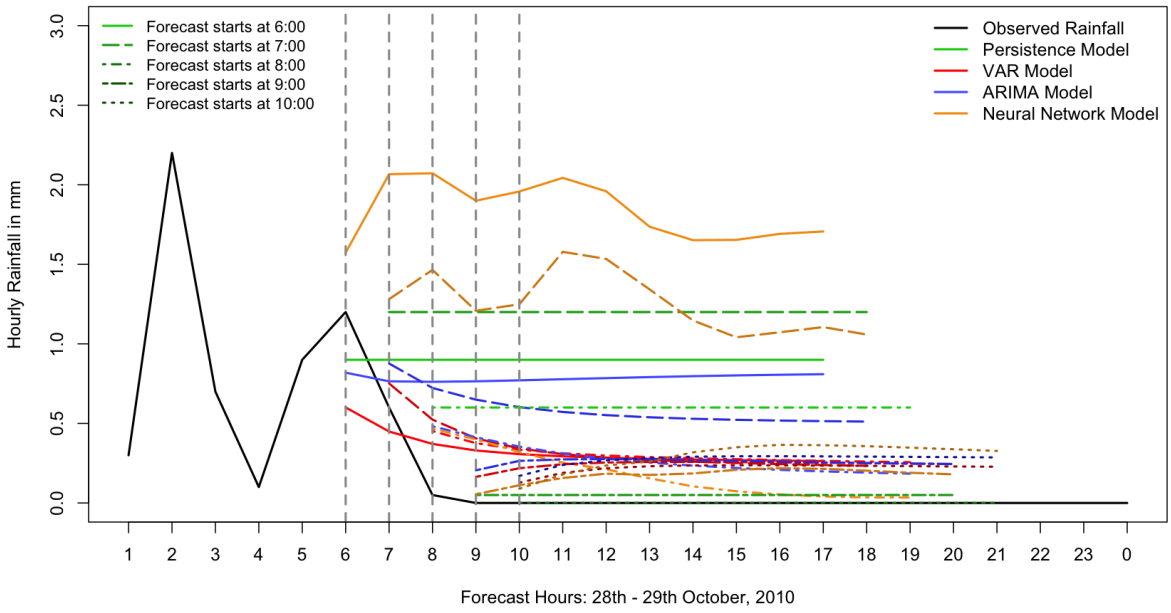

Figure 3.3: Case Study: Rain forecasts of different models when rain stops after a peak

Case 4: Rainfall persists throughout the period

In this case (11th - 12th January, 2010), rainfall persisting a few hours is considered as shown in Figure 3.4. The persistence model is quite inaccurate as the forecasts are flat and changing with every starting point. The VAR and the ARIMA models are quite consistent in their respective forecasts where some amount of rainfall is predicted. The Neural Network model forecasted a wave of rainfall that diminishes with time which is

not observed with the actual rainfall. Hence, the ARIMA model performed better than the other models.

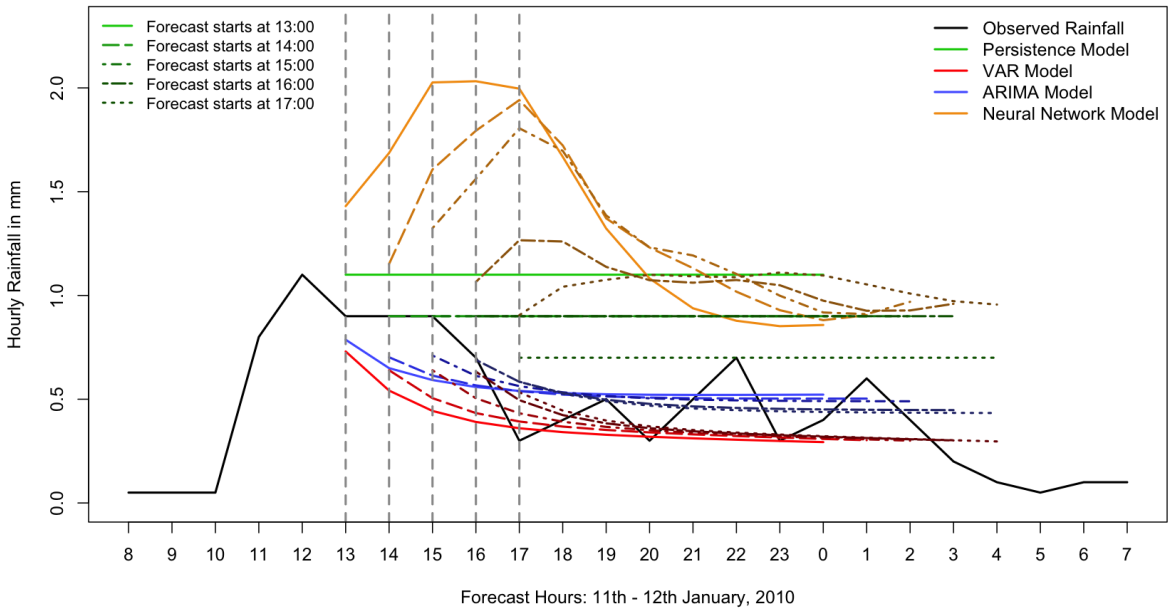

Figure 3.4: Case Study: Rain forecasts of different models when rainfall persists throughout the period

By these case studies, it follows that the persistence model is considerably bad at forecasting for the next 12 hours. Though Neural Network model performed well in some cases (Case 1 and Case 2), it forecasted poorly when high rainfall is observed currently. The VAR and the ARIMA models performed consistently well in the cases 1, 3 and 4.

3.2 Monthly Case Studies

To understand the behavior of models over a long period, monthly case studies are helpful. A considerably dry month (May, 2010) and a rainy month (January) are taken for this study. With the observed rainfall, the 1st hour, 3rd hour and 12th hour forecasts of all the models are plotted.

Case 1: January, 2010

Since January is comparatively rainy, lots of rainfall peaks are seen. Along with the observed rainfall peaks, the 1st hour forecasts are plotted in the Figure 3.5. Also, the mean rainfall of this period is depicted. The ARIMA Model and the VAR model forecasts (1st hour) are very close to the actual rainfall and the predictions are accurate. The Neural Network model over predicts the rainfall peaks in some cases.

In the Figure 3.6, the 3rd hour forecasts of all models are depicted along with the observed rainfall. The peaks become more spread and more cases of over-prediction are witnessed in Neural Network model. The ARIMA Model and the VAR model forecasts are close as before and the variation is to a larger extent.

Figure 3.7 shows the 12th hour forecasts of the models. It follows that not much variation in the ARIMA Model and the VAR model forecasts are seen compared to the 3rd hour forecasts. The peaks of Neural Network model become smaller compared to the 3rd hour forecasts and frequent peaks are seen.

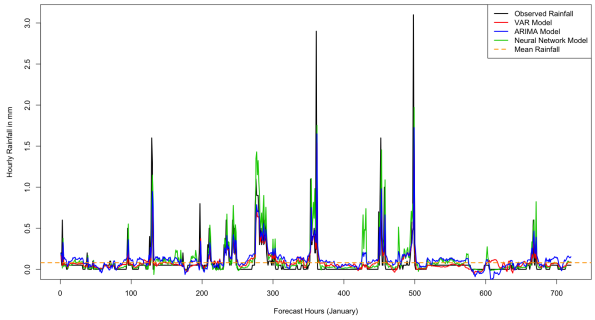

Figure 3.5: Case Study: 1st hour forecasts of different models in January, 2010

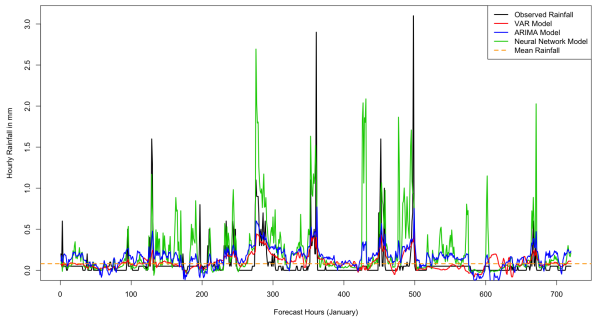

Figure 3.6: Case Study: 3rd hour forecasts of different models in January, 2010

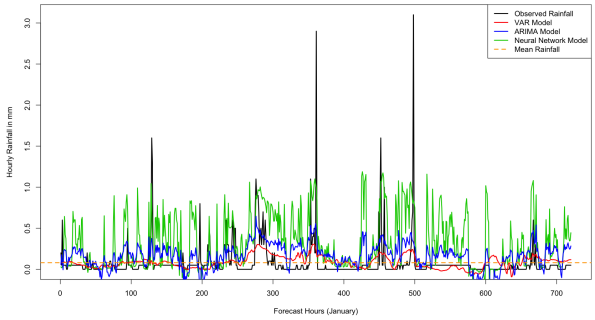

Figure 3.7: Case Study: 12th hour forecasts of different models in January, 2010

Case 2: May, 2010

A dry month, such as May, is also studied to know how the models forecast infrequent rainfall peaks. Very few rainfall peaks are seen in May, 2010. As before the 1st hour, 3rd hour and the 12th hour forecasts are plotted. Figure 3.8 shows the 1st hour forecasts with the mean rainfall in May, 2010. It can be seen that all the forecasts overlap with each other and the forecasts are considerably accurate. The ARIMA model seems to pick up the peaks in rainfall. All the models show very little variation in forecasts when the weather is dry. The VAR model is tightly overlapped with the actual rainfall.

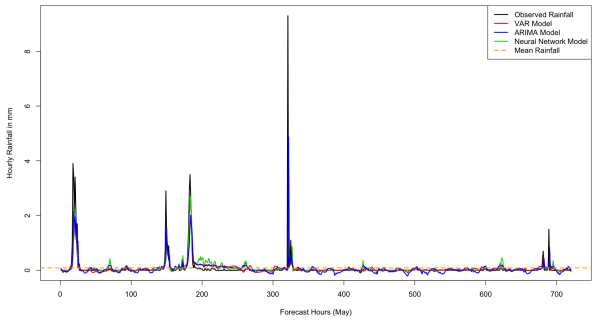

Figure 3.8: Case Study: 1st hour forecasts of different models in May, 2010

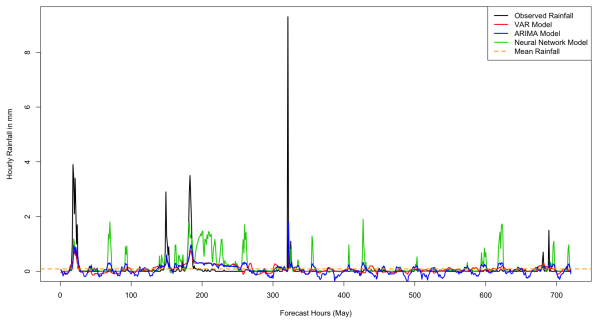

Figure 3.9: Case Study: 3rd hour forecasts of different models in May, 2010

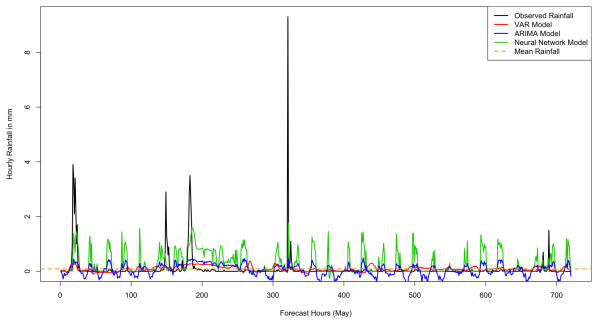

Figure 3.10: Case Study: 12th hour forecasts of different models in May, 2010

The 3rd hour forecasts are plotted in Figure 3.9. As before, the VAR model is tightly overlapped with the actual rainfall. The ARIMA model shows much variation than the 1st hour forecasts and the peaks are not accurately predicted. The Neural Network model forecasts show frequent spikes but the extent is not accurate. Figure 3.10 shows the 12th four forecasts. While the VAR model forecasts do not show much variation and are quite close to mean rainfall throughout the month, the ARIMA model forecasts show much variation compared to the 3rd hour forecasts. The Neural Network model forecasts show frequent peaks and are not accurate.

In essence, it is observed that as the forecast hours increase from 1st to 12th hour, the models become more and more inefficient. The ARIMA and the VAR models are accurate to some extent while the Neural Network model overpredicts the rainfall.

3.3 Results

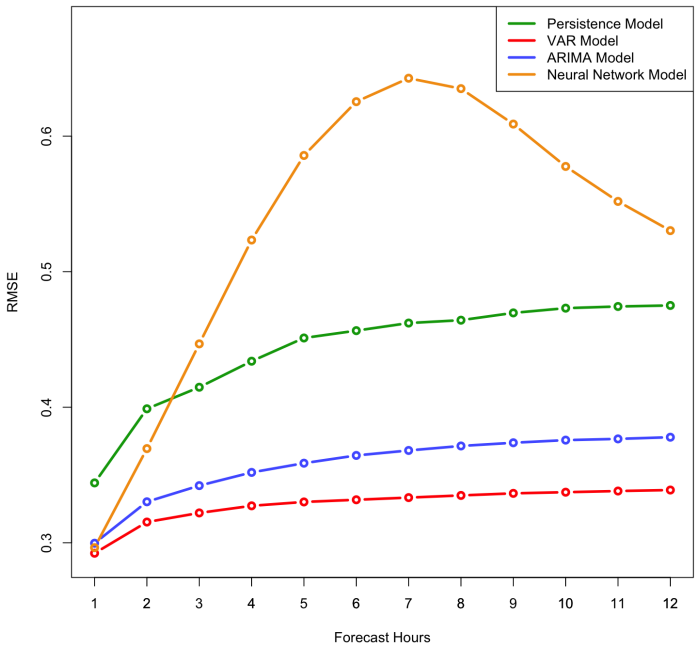

Figure 3.11: RMSE of different models at each forecast hour

The models are assessed by the RMSE values. The RMSE values of all the models are calculated for each of the forecast hour (1 hour to 12 hours). These values are plotted in Figure 3.11. It is evident from this figure that the persistence model is quite accurate at the first forecast hour, since it is very likely that the weather at the next hour will be the similar to the weather at the present hour. But as the rainfall is forecasted up to 12 hours ahead, the RMS Errors increase gradually. The Neural Network model shows abrupt changes in the RMSE values. Though the errors for the first two forecast hours are

lesser than the persistence model, at higher forecast hours, the errors become higher and the model performs poorly as compared to the persistence model. The ARIMA model performs much better than the persistence model through all the forecast hours and the increasing trend of errors with forecast hours is witnessed. The VAR model performs even better in all the forecast hours. The first hour RMS error is the lowest among all the models and the errors are almost flat from 3rd to 12th forecasts hours. Thus, the VAR model performs the best among all the models.

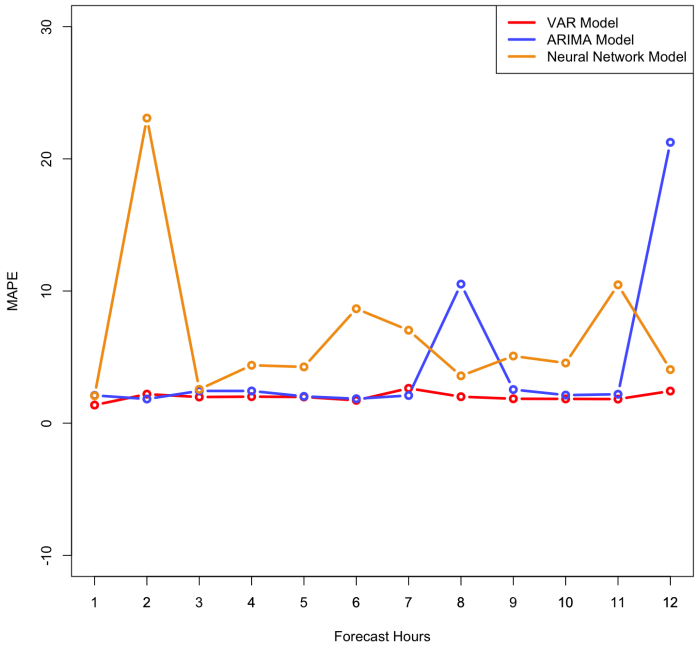

Figure 3.12: MAPE of different models at each forecast hour

Apart from the RMSE values, MAPE (Mean Absolute Percentage Error) values are also calculated for the 12 forecast models. The plot showing the MAPE values across 12 forecast hours for the ARIMA, the VAR and the Neural Network models is depicted in Figure 3.12. The VAR model shows consistent low errors across the 12 forecast hours. The ARIMA model shows slight high errors in 8th and 12th forecast hours. The Neural Network model shows inefficient MAPE values throughout and the 2nd forecast hour MAPE is quite high. So, it is evident that the VAR model performed consistently well by the MAPE values.

3.4 Skill Scores

Based on the average Mean Squared Errors (MSE) of the models, the skill scores are computed for the first three forecast hours and recorded in the Table 3.1. The persistence model is taken as the reference for computing the skill scores. So, the skill scores show

Model	**Skill Score**
Persistence Model	0
ARIMA Model	0.2972043
VAR Model	0.3581482
Neural Network Model	0.05694907

Table 3.1: Skill Scores of Models

how better or worse the given model performed as compared to the reference model. It is evident from the table that the Neural Network performed marginally better than the persistence model for the first three hours. While the ARIMA and the VAR models performed better than the persistence model, the VAR model has the highest skill score of 0.358 when average MSE are considered.

Chapter 4

Conclusion

Several insights are established from this study. Regarding the variables that influence the rainfall the most, the surface level parameters (Pressure, Temperature, Dew Point, Wind speed and Wind Direction) and the Specific Humidity of the 5 different pressure levels above the surface (750, 1500, 3000, 5500 and 12000 meters above the surface) greatly influence future rainfall. The hourly rainfall modelled as time series is found to be stationary, which help to model the series using the mentioned methods. The models are fitted with expanding window method and the 12 hour forecasts are predicted by each of the models for the year 2010. In any forecasting method, the first hour forecast is highly accurate compared to the later hour forecasts. As the forecast hour increases from the 1st hour to the 12th hour, the accuracy of the models gradually decrease. This is mainly because the models revert towards the mean rainfall of the period from any given rainfall level.

The time series data are fitted with the VAR, the ARIMA and the Neural Network models. By the hourly case studies, it is evident that after a peak rainfall, the forecasts of the ARIMA model and the VAR model correctly tend to revert towards the mean. The imminent peaks were forecasted well by the Neural Network model. By the monthly case studies, it is found that the 1st hour forecasts are accurate than the 3rd hour forecasts which are in turn accurate than the 12th hour forecasts of all models. The RMSE and MAPE are used as the model assessment metrics. By the RMSE values across 12 forecast hours, the Neural Network model gave inefficient results while the ARIMA and the VAR models gave consistently accurate results. By the MAPE values, the VAR model gave consistently precise results and the ARIMA model gave fairly accurate values. The Neural Network model gave inconsistent results throughout the 12 forecast hours. The skill scores of the models are computed with the persistence model as the baseline model. The average MSE of 3 forecasts are considered in calculation of the skill scores, since short term rain forecast is desired. The VAR and the ARIMA models have skill scores of 0.358 and 0.297 respectively suggesting that these models performed better than the persistence model. The Neural Network model has the skill score of 0.05, suggesting there is slight

improvement compared to the persistence model.

Finally, on examination of accuracy metrics such as, RMSE, MAPE and skill score values, and the hourly and monthly case studies, it is found that the VAR model is the most accurate model, closely followed by the ARIMA model. Though Neural Network model took the last place in terms of the accuracy, it is better for accurately forecasting imminent peaks in rainfall suggesting that the machine learning techniques have a potential role in forecasting extreme rainfall in the next hour.

Appendix A

Acronyms and Units

VAR: Vector AutoRegression
ARIMA: AutoRegressive Integrated Moving Average
RMSE: Root Mean Square Error
MAPE: Mean Absolute Percentage Error
NWP: Numerical Weather Prediction
ADF: Augmented Dickey Fuller
KPSS: Kwiatkowski–Phillips–Schmidt–Shin
ACF: Auto Correlation Function
PACF: Partial Auto-Correlation Function
MSE: Mean Squared Error
hPa: hectopascals (1 hPa = 100 pascals)
knot: Unit of speed equal to one nautical mile per hour (1.852 km/h)

Appendix B

Code

```
# Loading required libraries
library(forecast) # ARIMA Model, Neural Network Model and forecasting
library(vars) # VAR Model
library(greybox) # Accuracy Metrics
library(tseries) # Time Series Functions

setwd("/ABHI/Workspace/Study/UCD/SEM_3/RESEARCH_PROJECT/R")

# Loading the dataset
FullData = read.csv("FullData_Dublin_Airport.csv")

### PRELIMINARY ANALYSIS ###
Rain = ts(FullData$Rainfall, start = c(1981, 1), end = c(2015, 8760), frequency =
    ↪ 365.25*24)

png(file="Plots/RainTimeSeries.png",width=2100,height=900,res=200)
plot(Rain, ylab = "Hourly Rainfall (in mm)", col = "#046997")
dev.off()

png(file="Plots/RainDensity.png",width=1500,height=1200,res=300)
plot(density(log(Rain)), col = "#5264FF", xlab = "Log - Rainfall", main = "", lwd
    ↪  = 2)
dev.off()

png(file="Plots/Decomposition.png",width=2000,height=1400,res=200)
fit = decompose(Rain, "multiplicative")
plot(fit, col = "#046997")
```

```
dev.off()

# Augmented Dickey Fuller Test
adf.test(Rain)

# KPSS Test for Stationarity
kpss.test(Rain)

mylm = tslm(Rainfall~ Pressure + Air_Temp + Dew_Point + Wind_direction + Wind_
    ↪ speed + X200.Temperature + X200.Wind_direction + X200.Wind_speed +
    ↪ X200.Specific_Humidity + X500.Temperature + X500.Wind_direction + X500.
    ↪ Wind_speed + X500.Specific_Humidity + X700.Temperature + X700.Wind_
    ↪ direction + X700.Wind_speed + X700.Specific_Humidity + X850.Temperature
    ↪ + X850.Wind_direction + X850.Wind_speed + X850.Specific_Humidity + X925.
    ↪ Temperature + X925.Wind_direction + X925.Wind_speed + X925.Specific_
    ↪ Humidity, data = ts(FullData[1:8760,]))
summary(mylm)

### MODEL FITTING ###
# Initializing matrices to store results of different models
resmat=matrix(ncol = 24)
resdf = data.frame(resmat)
naive.resdf = resdf
arima.resdf = resdf
var.resdf = resdf
neuralnet.resdf = resdf

# Setting up time series data
horizon = 12
ts.rain = ts(FullData$Rainfall, frequency = 8766, start = c(1981,1))
test <- window(ts.rain,start=2010, end = c(2010, 8766))
n <- length(test) - h + 1
data.ts = ts(FullData[,c(2,3,4,5,6,11,15,19,23,27)], frequency = 8766, start = c
    ↪ (1981,1))

# Model Initializations
train <- window(ts.rain,end=1981.99999)
init.xreg <- window(data.ts, end=1981.99999)
var.data.ts = ts(FullData[,c(2,3,4,5,6,7,11,15,19,23,27)], frequency = 8766, start = c
```

```
    ↪ (1981,1))
arima.fit = auto.arima(train, xreg = init.xreg, optim.method = "CG")
neuralnet.fit <- nnetar(train, xreg = init.xreg)

# Expanding window Iterations
iter = 0
i = 1
while (i < n) {
  # Training and test windows
  x <- window(ts.rain, end=2009.99999 + (i-1)/8766)
  y = window(ts.rain, start = (2009.99999 + (i-1)/8766), end = (0.00128 +
      ↪ 2009.99999 + (i-1)/8766))
  cur.xreg.window = window(data.ts, end=2009.99999 + (i-1)/8766)
  xreg.window = window(data.ts, start=c(2009,1),end=2009.99999 + (i-1)/8766)

  # Persistence Model
  naive.fit = naive(x, h = horizon)
  naive.resdf[i,1:12] = y
  naive.resdf[i,13:24] = naive.fit$mean

  # VAR Model
  var.cur.xreg.window = window(var.data.ts, end=2009.99999 + (i-1)/8766)
  var.fit = VAR(var.cur.xreg.window, type = "both")
  var.forecast = forecast(var.fit, h = horizon)
  var.resdf[i,1:12] = y
  var.resdf[i,13:24] = var.forecast$forecast$Rainfall$mean

  fc.c1 <- forecast(xreg.window[,1], h = horizon)
  fc.c2 <- forecast(xreg.window[,2], h = horizon)
  fc.c3 <- forecast(xreg.window[,3], h = horizon)
  fc.c4 <- forecast(xreg.window[,4], h = horizon)
  fc.c5 <- forecast(xreg.window[,5], h = horizon)
  fc.c6 <- forecast(xreg.window[,6], h = horizon)
  fc.c7 <- forecast(xreg.window[,7], h = horizon)
  fc.c8 <- forecast(xreg.window[,8], h = horizon)
  fc.c9 <- forecast(xreg.window[,9], h = horizon)
  fc.c10 <- forecast(xreg.window[,10], h = horizon)
  newxreg <- as.matrix(cbind(fc.c1$mean, fc.c2$mean, fc.c3$mean, fc.c4$mean,
      ↪ fc.c5$mean,
```

```
                                    fc.c6$mean, fc.c7$mean, fc.c8$mean, fc.c9$mean, fc.
                                        ↪ c10$mean))
  colnames(newxreg) = colnames(data.ts)

  # ARIMA Model
  arima.refit <- Arima(x, xreg = cur.xreg.window, optim.method = "CG", model
      ↪ = arima.fit)
  arima.forecast = forecast(arima.refit, h = horizon, xreg = newxreg)
  arima.resdf[i,1:12] = y
  arima.resdf[i,13:24] = arima.forecast$mean

  # Time Series Neural Network Model
  neuralnet.refit <- nnetar(x, model = neuralnet.fit, xreg = cur.xreg.window, scale.
      ↪ inputs = TRUE)
  neuralnet.forecast = forecast(neuralnet.refit, h = horizon, xreg = newxreg)
  neuralnet.resdf[i,1:12] = y
  neuralnet.resdf[i,13:24] = neuralnet.forecast$mean

  i=i+1
  iter = iter + 1
  print(iter)
}

### PLOTS ###
# Hourly Case Studies
# Case 1: Rain -> Rain
png(file="HRForecastRR.png",width=1700,height=1000,res=150)
plot(naive.resdf[272:295,1], type = "l", col = 1, lwd = 2, ylim = c(0,2.3),
     xaxt='n', ylab = "Hourly Rainfall in mm", xlab = "Forecast Hours: 11th - 12th
         ↪  January, 2010")
lines(6:17, naive.resdf[277,13:24], lwd = 2, col = "#05CE0F")
lines(6:17, var.resdf[277, 13:24], lwd = 2, col = "#FF0000")
lines(6:17, arima.resdf[277, 13:24], lwd = 2, col = "#5264FF")
lines(6:17, neuralnet.resdf[277, 13:24], lwd = 2, col = "#F39C12")
abline(v = 6, col = "gray60", lty=2, lwd = 2)

lines(7:18, naive.resdf[278,13:24], lwd = 2, col = "#04A40C", lty = 5)
lines(7:18, var.resdf[278, 13:24], lwd = 2, col = "#DA0202", lty = 5)
lines(7:18, arima.resdf[278, 13:24], lwd = 2, col = "#3749E5", lty = 5)
```

```
lines(7:18, neuralnet.resdf[278, 13:24], lwd = 2, col = "#D68910", lty = 5)
abline(v = 7, col = "gray60", lty=2, lwd = 2)

lines(8:19, naive.resdf[279,13:24], lwd = 2, col = "#038109", lty = 4)
lines(8:19, var.resdf[279, 13:24], lwd = 2, col = "#AC0202", lty = 4)
lines(8:19, arima.resdf[279, 13:24], lwd = 2, col = "#1A29AA", lty = 4)
lines(8:19, neuralnet.resdf[279, 13:24], lwd = 2, col = "#B9770E", lty = 4)
abline(v = 8, col = "gray60", lty=2, lwd = 2)

lines(9:20, naive.resdf[280,13:24], lwd = 2, col = "#0A6101", lty = 6)
lines(9:20, var.resdf[280, 13:24], lwd = 2, col = "#780202", lty = 6)
lines(9:20, arima.resdf[280, 13:24], lwd = 2, col = "#2C3478", lty = 6)
lines(9:20, neuralnet.resdf[280, 13:24], lwd = 2, col = "#9C640C", lty = 6)
abline(v = 9, col = "gray60", lty=2, lwd = 2)

lines(10:21, naive.resdf[281,13:24], lwd = 2, col = "#0A6101", lty = 3)
lines(10:21, var.resdf[281, 13:24], lwd = 2, col = "#780202", lty = 3)
lines(10:21, arima.resdf[281, 13:24], lwd = 2, col = "#2C3478", lty = 3)
lines(10:21, neuralnet.resdf[281, 13:24], lwd = 2, col = "#7E5109", lty = 3)
abline(v = 10, col = "gray60", lty=2, lwd = 2)

legend("topright", c("Observed Rainfall", "Persistence Model", "VAR Model", "
    ↪ ARIMA Model", "Neural Network Model"), lwd = 2,
       col = c(1, "#05CE0F", "#FF0000", "#5264FF", "#F39C12"), bty = "n")
legend("topleft", c("Forecast starts at 13:00", "Forecast starts at 14:00",
                    "Forecast starts at 15:00", "Forecast starts at 16:00", "Forecast
                        ↪ starts at 17:00"),
       lwd = 2, lty = c(1, 5, 4, 6, 3), col = c("#05CE0F", "#04A40C", "#038109", "
           ↪ #0A6101", "#0A6101"), bty = "n", cex = 0.9)
axis(1, at = 1:24, labels = c(8:23,0:7))
dev.off()


# Case 2: No Rain -> No Rain
png(file="HRForecastNN.png",width=1700,height=1000,res=150)
plot(naive.resdf[8:31,1], type = "l", col = 1, lwd = 2, ylim = c(0,1),
     xaxt='n', ylab = "Hourly Rainfall in mm", xlab = "Forecast Hours: 1st - 2nd
         ↪ January, 2010")
lines(6:17, naive.resdf[13,13:24], lwd = 2, col = "#04A40C")
```

```
lines(6:17, var.resdf[13, 13:24], lwd = 2, col = "#DA0202")
lines(6:17, arima.resdf[13, 13:24], lwd = 2, col = "#3749E5")
lines(6:17, neuralnet.resdf[13, 13:24], lwd = 2, col = "#D68910")
abline(v = 6, col = "gray60", lty=2, lwd = 2)

lines(7:18, naive.resdf[14,13:24], lwd = 2, col = "#04A40C", lty = 5)
lines(7:18, var.resdf[14, 13:24], lwd = 2, col = "#DA0202", lty = 5)
lines(7:18, arima.resdf[14, 13:24], lwd = 2, col = "#3749E5", lty = 5)
lines(7:18, neuralnet.resdf[14, 13:24], lwd = 2, col = "#D68910", lty = 5)
abline(v = 7, col = "gray60", lty=2, lwd = 2)

lines(8:19, naive.resdf[15,13:24], lwd = 2, col = "#05CE0F", lty = 4)
lines(8:19, var.resdf[15, 13:24], lwd = 2, col = "#FF0000", lty = 4)
lines(8:19, arima.resdf[15, 13:24], lwd = 2, col = "#5264FF", lty = 4)
lines(8:19, neuralnet.resdf[15, 13:24], lwd = 2, col = "#F39C12", lty = 4)
abline(v = 8, col = "gray60", lty=2, lwd = 2)

lines(9:20, naive.resdf[16,13:24], lwd = 2, col = "#04A40C", lty = 6)
lines(9:20, var.resdf[16, 13:24], lwd = 2, col = "#DA0202", lty = 6)
lines(9:20, arima.resdf[16, 13:24], lwd = 2, col = "#3749E5", lty = 6)
lines(9:20, neuralnet.resdf[16, 13:24], lwd = 2, col = "#D68910", lty = 6)
abline(v = 9, col = "gray60", lty=2, lwd = 2)

lines(10:21, naive.resdf[17,13:24], lwd = 2, col = "#038109", lty = 3)
lines(10:21, var.resdf[17, 13:24], lwd = 2, col = "#AC0202", lty = 3)
lines(10:21, arima.resdf[17, 13:24], lwd = 2, col = "#1A29AA", lty = 3)
lines(10:21, neuralnet.resdf[17, 13:24], lwd = 2, col = "#B9770E", lty = 3)
abline(v = 10, col = "gray60", lty=2, lwd = 2)

legend("topright", c("Observed Rainfall", "Persistence Model", "VAR Model", "
    ↪ ARIMA Model", "Neural Network Model"), lwd = 2,
       col = c(1, "#05CE0F", "#FF0000", "#5264FF", "#F39C12"), bty = "n")
legend("topleft", c("Forecast starts at 13:00", "Forecast starts at 14:00",
                    "Forecast starts at 15:00", "Forecast starts at 16:00", "Forecast
                        ↪ starts at 17:00"),
       lwd = 2, lty = c(1, 5, 4, 6, 3), col = c("#05CE0F", "#04A40C", "#038109", "
           ↪ #0A6101", "#0A6101"), bty = "n", cex = 0.9)
axis(1, at = 1:24, labels = c(8:23,0:7))
dev.off()
```

```
# Case 3: NoRain -> Rain
png(file="HRForecastNR.png",width=1700,height=1000,res=150)
plot(naive.resdf[2078:2101,1], type = "l", col = 1, lwd = 2, xaxt='n', ylim = c(0, 3.2),
    ↪ ylab = "Hourly Rainfall in mm", xlab = "Forecast Hours: 27th - 28th March,
    ↪ 2010")
lines(6:17, naive.resdf[2083,13:24], lwd = 2, col = "#05CE0F")
lines(6:17, var.resdf[2083, 13:24], lwd = 2, col = "#FF0000")
lines(6:17, arima.resdf[2083, 13:24], lwd = 2, col = "#5264FF")
lines(6:17, neuralnet.resdf[2083, 13:24], lwd = 2, col = "#F39C12")
abline(v = 6, col = "gray60", lty=2, lwd = 2)

lines(7:18, naive.resdf[2084,13:24], lwd = 2, col = "#04A40C", lty = 5)
lines(7:18, var.resdf[2084, 13:24], lwd = 2, col = "#DA0202", lty = 5)
lines(7:18, arima.resdf[2084, 13:24], lwd = 2, col = "#3749E5", lty = 5)
lines(7:18, neuralnet.resdf[2084, 13:24], lwd = 2, col = "#D68910", lty = 5)
abline(v = 7, col = "gray60", lty=2, lwd = 2)

lines(8:19, naive.resdf[2085,13:24], lwd = 2, col = "#05CE0F", lty = 4)
lines(8:19, var.resdf[2085, 13:24], lwd = 2, col = "#FF0000", lty = 4)
lines(8:19, arima.resdf[2085, 13:24], lwd = 2, col = "#5264FF", lty = 4)
lines(8:19, neuralnet.resdf[2085, 13:24], lwd = 2, col = "#F39C12", lty = 4)
abline(v = 8, col = "gray60", lty=2, lwd = 2)

lines(9:20, naive.resdf[2086,13:24], lwd = 2, col = "#04A40C", lty = 6)
lines(9:20, var.resdf[2086, 13:24], lwd = 2, col = "#DA0202", lty = 6)
lines(9:20, arima.resdf[2086, 13:24], lwd = 2, col = "#3749E5", lty = 6)
lines(9:20, neuralnet.resdf[2086, 13:24], lwd = 2, col = "#D68910", lty = 6)
abline(v = 9, col = "gray60", lty=2, lwd = 2)

lines(10:21, naive.resdf[2087,13:24], lwd = 2, col = "#038109", lty = 3)
lines(10:21, var.resdf[2087, 13:24], lwd = 2, col = "#AC0202", lty = 3)
lines(10:21, arima.resdf[2087, 13:24], lwd = 2, col = "#1A29AA", lty = 3)
lines(10:21, neuralnet.resdf[2087, 13:24], lwd = 2, col = "#B9770E", lty = 3)
abline(v = 10, col = "gray60", lty=2, lwd = 2)

legend("topright", c("Observed Rainfall", "Persistence Model", "VAR Model", "
    ↪ ARIMA Model", "Neural Network Model"), lwd = 2,
```

```
        col = c(1, "#05CE0F", "#FF0000", "#5264FF", "#F39C12"), bty = "n")
legend("topleft", c("Forecast starts at 19:00", "Forecast starts at 20:00", "Forecast
    ↪ starts at 21:00", "Forecast starts at 22:00", "Forecast starts at 23:00"),
        lwd = 2, lty = c(1, 5, 4, 6, 3), col = c("#05CE0F", "#04A40C", "#038109", "
            ↪ #0A6101", "#0A6101"), bty = "n", cex = 0.9)
axis(1, at = 1:24, labels = c(14:23, 0:13))
dev.off()


# Case 4: Rain -> No Rain
png(file="HRForecastRN.png",width=1700,height=1000,res=150)
plot(naive.resdf[7229:7252,1], type ="l", col = 1, lwd = 2, xaxt='n', ylim = c(0,3.0),
    ↪ ylab = "Hourly Rainfall in mm", xlab = "Forecast Hours: 28th - 29th
    ↪ October, 2010")
lines(6:17, naive.resdf[7234,13:24], lwd = 2, col = "#05CE0F")
lines(6:17, var.resdf[7234, 13:24], lwd = 2, col = "#FF0000")
lines(6:17, arima.resdf[7234, 13:24], lwd = 2, col = "#5264FF")
lines(6:17, neuralnet.resdf[7234, 13:24], lwd = 2, col = "#F39C12")
abline(v = 6, col = "gray60", lty=2, lwd = 2)

lines(7:18, naive.resdf[7235,13:24], lwd = 2, col = "#04A40C", lty = 5)
lines(7:18, var.resdf[7235, 13:24], lwd = 2, col = "#DA0202", lty = 5)
lines(7:18, arima.resdf[7235, 13:24], lwd = 2, col = "#3749E5", lty = 5)
lines(7:18, neuralnet.resdf[7235, 13:24], lwd = 2, col = "#D68910", lty = 5)
abline(v = 7, col = "gray60", lty=2, lwd = 2)

lines(8:19, naive.resdf[7236,13:24], lwd = 2, col = "#05CE0F", lty = 4)
lines(8:19, var.resdf[7236, 13:24], lwd = 2, col = "#FF0000", lty = 4)
lines(8:19, arima.resdf[7236, 13:24], lwd = 2, col = "#5264FF", lty = 4)
lines(8:19, neuralnet.resdf[7236, 13:24], lwd = 2, col = "#F39C12", lty = 4)
abline(v = 8, col = "gray60", lty=2, lwd = 2)

lines(9:20, naive.resdf[7237,13:24], lwd = 2, col = "#04A40C", lty = 6)
lines(9:20, var.resdf[7237, 13:24], lwd = 2, col = "#DA0202", lty = 6)
lines(9:20, arima.resdf[7237, 13:24], lwd = 2, col = "#3749E5", lty = 6)
lines(9:20, neuralnet.resdf[7237, 13:24], lwd = 2, col = "#D68910", lty = 6)
abline(v = 9, col = "gray60", lty=2, lwd = 2)

lines(10:21, naive.resdf[7238,13:24], lwd = 2, col = "#038109", lty = 3)
```

```
lines(10:21, var.resdf[7238, 13:24], lwd = 2, col = "#AC0202", lty = 3)
lines(10:21, arima.resdf[7238, 13:24], lwd = 2, col = "#1A29AA", lty = 3)
lines(10:21, neuralnet.resdf[7238, 13:24], lwd = 2, col = "#B9770E", lty = 3)
abline(v = 10, col = "gray60", lty=2, lwd = 2)

legend("topright", c("Observed Rainfall", "Persistence Model", "VAR Model", "
    ↪ ARIMA Model", "Neural Network Model"), lwd = 2,
        col = c(1, "#05CE0F", "#FF0000", "#5264FF", "#F39C12"), bty = "n")
legend("topleft", c("Forecast starts at 6:00", "Forecast starts at 7:00", "Forecast
    ↪ starts at 8:00", "Forecast starts at 9:00", "Forecast starts at 10:00"),
        lwd = 2, lty = c(1, 5, 4, 6, 3), col = c("#05CE0F", "#04A40C", "#038109", "
            ↪ #0A6101", "#0A6101"), bty = "n", cex = 0.9)
axis(1, at = 1:24, labels = c(1:23, 0))
dev.off()


# Monthly Case Studies
### January
png(file="Jan1.png",width=1700,height=1000,res=125)
rain.wind = window(ts.rain, start = 2010, end = c(2010,720))
plot(1:720, rain.wind, type = "l", col = 1, lwd = 2, ylab = "Hourly Rainfall in mm",
    ↪ xlab = "Forecast Hours (January)")
lines(1:720, neuralnet.resdf[1:720,13], type = "l", col = 3, lwd = 2)
lines(1:720, var.resdf[1:720,13], type = "l", col = "#FF0000", lwd = 2)
lines(1:720, arima.resdf[1:720,13], type = "l", col = "#0000FF", lwd = 2)
meanmy = mean(rain.wind)
abline(h = meanmy, col = "orange", lwd = 2, lty = 2)
legend("topright", c("Observed Rainfall", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model", "Mean Rainfall"), lwd = 2,
        col = c(1, "#FF0000", "#0000FF", 3, "orange"), lty = c(1,1,1,1,2))
dev.off()

png(file="Jan3.png",width=1700,height=1000,res=125)
rain.wind = window(ts.rain, start = 2010, end = c(2010,720))
plot(1:720, rain.wind, type = "l", col = 1, lwd = 2, ylab = "Hourly Rainfall in mm",
    ↪ xlab = "Forecast Hours (January)")
lines(1:720, neuralnet.resdf[1:720,15], type = "l", col = 3, lwd = 2)
lines(1:720, var.resdf[1:720,15], type = "l", col = "#FF0000", lwd = 2)
lines(1:720, arima.resdf[1:720,15], type = "l", col = "#0000FF", lwd = 2)
```

```
abline(h = meanmy, col = "orange", lwd = 2, lty = 2)
legend("topright", c("Observed Rainfall", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model", "Mean Rainfall"), lwd = 2,
        col = c(1, "#FF0000", "#0000FF", 3, "orange"), lty = c(1,1,1,1,2))
dev.off()

png(file="Jan12.png",width=1700,height=1000,res=125)
rain.wind = window(ts.rain, start = 2010, end = c(2010,720))
plot(1:720, rain.wind, type = "l", col = 1, lwd = 2, ylab = "Hourly Rainfall in mm",
    ↪ xlab = "Forecast Hours (January)")
lines(1:720, neuralnet.resdf[1:720,24], type = "l", col = 3, lwd = 2)
lines(1:720, var.resdf[1:720,24], type = "l", col = "#FF0000", lwd = 2)
lines(1:720, arima.resdf[1:720,24], type = "l", col = "#0000FF", lwd = 2)
abline(h = meanmy, col = "orange", lwd = 2, lty = 2)
legend("topright", c("Observed Rainfall", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model", "Mean Rainfall"), lwd = 2,
        col = c(1, "#FF0000", "#0000FF", 3, "orange"), lty = c(1,1,1,1,2))
dev.off()

### May
png(file="May1.png",width=1700,height=1000,res=125)
rain.wind = window(ts.rain, start = c(2010,3601), end = c(2010,4320))
plot(1:720, rain.wind, type = "l", col = 1, lwd = 2, ylab = "Hourly Rainfall in mm",
    ↪ xlab = "Forecast Hours (May)")
lines(1:720, neuralnet.resdf[3601:4320,13], type = "l", col = 3, lwd = 2)
lines(1:720, var.resdf[3601:4320,13], type = "l", col = "#FF0000", lwd = 2)
lines(1:720, arima.resdf[3601:4320,13], type = "l", col = "#0000FF", lwd = 2)
meanmy = mean(rain.wind)
abline(h = meanmy, col = "orange", lwd = 2, lty = 2)
legend("topright", c("Observed Rainfall", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model", "Mean Rainfall"), lwd = 2,
        col = c(1, "#FF0000", "#0000FF", 3, "orange"), lty = c(1,1,1,1,2))
dev.off()

png(file="May3.png",width=1700,height=1000,res=125)
rain.wind = window(ts.rain, start = c(2010,3601), end = c(2010,4320))
plot(1:720, rain.wind, type = "l", col = 1, lwd = 2, ylab = "Hourly Rainfall in mm",
    ↪ xlab = "Forecast Hours (May)")
lines(1:720, neuralnet.resdf[3601:4320,15], type = "l", col = 3, lwd = 2)
```

```
lines(1:720, var.resdf[3601:4320,15], type = "l", col = "#FF0000", lwd = 2)
lines(1:720, arima.resdf[3601:4320,15], type = "l", col = "#0000FF", lwd = 2)
abline(h = meanmy, col = "orange", lwd = 2, lty = 2)
legend("topright", c("Observed Rainfall", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model", "Mean Rainfall"), lwd = 2,
        col = c(1, "#FF0000", "#0000FF", 3, "orange"), lty = c(1,1,1,1,2))
dev.off()

png(file="May12.png",width=1700,height=1000,res=125)
rain.wind = window(ts.rain, start = c(2010,3601), end = c(2010,4320))
plot(1:720, rain.wind, type = "l", col = 1, lwd = 2, ylab = "Hourly Rainfall in mm",
    ↪ xlab = "Forecast Hours (May)")
lines(1:720, neuralnet.resdf[3601:4320,24], type = "l", col = 3, lwd = 2)
lines(1:720, var.resdf[3601:4320,24], type = "l", col = "#FF0000", lwd = 2)
lines(1:720, arima.resdf[3601:4320,24], type = "l", col = "#0000FF", lwd = 2)
abline(h = meanmy, col = "orange", lwd = 2, lty = 2)
legend("topright", c("Observed Rainfall", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model", "Mean Rainfall"), lwd = 2,
        col = c(1, "#FF0000", "#0000FF", 3, "orange"), lty = c(1,1,1,1,2))
dev.off()

#### ACCURACY ####
Naive.Hour.Accuracy = data.frame()
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X13, naive.
    ↪ resdf$X1))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X14, naive.
    ↪ resdf$X2))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X15, naive.
    ↪ resdf$X3))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X16, naive.
    ↪ resdf$X4))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X17, naive.
    ↪ resdf$X5))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X18, naive.
    ↪ resdf$X6))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X19, naive.
    ↪ resdf$X7))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X20, naive.
    ↪ resdf$X8))
```

```
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X21, naive.
    ↪ resdf$X9))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X22, naive.
    ↪ resdf$X10))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X23, naive.
    ↪ resdf$X11))
Naive.Hour.Accuracy = rbind(Naive.Hour.Accuracy, accuracy(naive.resdf$X24, naive.
    ↪ resdf$X12))

Arima.Hour.Accuracy = data.frame()
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X13,
    ↪ arima.resdf$X1))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X14,
    ↪ arima.resdf$X2))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X15,
    ↪ arima.resdf$X3))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X16,
    ↪ arima.resdf$X4))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X17,
    ↪ arima.resdf$X5))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X18,
    ↪ arima.resdf$X6))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X19,
    ↪ arima.resdf$X7))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X20,
    ↪ arima.resdf$X8))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X21,
    ↪ arima.resdf$X9))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X22,
    ↪ arima.resdf$X10))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X23,
    ↪ arima.resdf$X11))
Arima.Hour.Accuracy = rbind(Arima.Hour.Accuracy, accuracy(arima.resdf$X24,
    ↪ arima.resdf$X12))

VAR.Hour.Accuracy = data.frame()
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X13, var.resdf$
    ↪ X1))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X14, var.resdf$
```

```
    ↪ X2))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X15, var.resdf$
    ↪ X3))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X16, var.resdf$
    ↪ X4))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X17, var.resdf$
    ↪ X5))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X18, var.resdf$
    ↪ X6))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X19, var.resdf$
    ↪ X7))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X20, var.resdf$
    ↪ X8))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X21, var.resdf$
    ↪ X9))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X22, var.resdf$
    ↪ X10))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X23, var.resdf$
    ↪ X11))
VAR.Hour.Accuracy = rbind(VAR.Hour.Accuracy, accuracy(var.resdf$X24, var.resdf$
    ↪ X12))

NeuralNet.Hour.Accuracy = data.frame()
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X13, neuralnet.resdf$X1))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X14, neuralnet.resdf$X2))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X15, neuralnet.resdf$X3))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X16, neuralnet.resdf$X4))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X17, neuralnet.resdf$X5))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X18, neuralnet.resdf$X6))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X19, neuralnet.resdf$X7))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X20, neuralnet.resdf$X8))
```

```
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X21, neuralnet.resdf$X9))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X22, neuralnet.resdf$X10))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X23, neuralnet.resdf$X11))
NeuralNet.Hour.Accuracy = rbind(NeuralNet.Hour.Accuracy, accuracy(neuralnet.resdf
    ↪ $X24, neuralnet.resdf$X12))

# For MAPE, MAPE() function is used instead of accuracy()

png(file="AccuracyRMSE.png",width=1500,height=1500,res=175)
plot(Naive.Hour.Accuracy$RMSE, type = "b", ylim = c(0.29,0.68), col = "#04A40C"
    ↪ , lwd = 3, xlab = "Forecast Hours", ylab = "RMSE")
lines(Arima.Hour.Accuracy$RMSE, type = "b", col = "#5264FF", lwd = 3)
lines(VAR.Hour.Accuracy$RMSE, type = "b", col = "#FF0000", lwd = 3)
lines(NeuralNet.Hour.Accuracy$RMSE, type = "b", col = "#F39C12", lwd = 3)
legend("topright", c("Persistence Model", "VAR Model", "ARIMA Model", "Neural
    ↪ Network Model"), lwd = 3,
       col = c("#04A40C", "#FF0000", "#5264FF", "#F39C12"))
axis(1, at = 1:12, labels = c(1:12))
dev.off()

# Skill Scores
Naive.Hour.Accuracy.MSE = mean(Naive.Hour.Accuracy$RMSE[1:3] ^ 2)
Arima.Hour.Accuracy.MSE = mean(Arima.Hour.Accuracy$RMSE[1:3] ^ 2)
VAR.Hour.Accuracy.MSE = mean(VAR.Hour.Accuracy$RMSE[1:3] ^ 2)
NeuralNet.Hour.Accuracy.MSE = mean(NeuralNet.Hour.Accuracy$RMSE[1:3] ^ 2)

SS.Naive = (Naive.Hour.Accuracy.MSE - Naive.Hour.Accuracy.MSE) / (0 - Naive.
    ↪ Hour.Accuracy.MSE)
SS.Arima = (Arima.Hour.Accuracy.MSE - Naive.Hour.Accuracy.MSE) / (0 - Naive.
    ↪ Hour.Accuracy.MSE)
SS.VAR = (VAR.Hour.Accuracy.MSE - Naive.Hour.Accuracy.MSE) / (0 - Naive.
    ↪ Hour.Accuracy.MSE)
SS.NeuralNet = (NeuralNet.Hour.Accuracy.MSE - Naive.Hour.Accuracy.MSE) / (0 -
    ↪ Naive.Hour.Accuracy.MSE)
```

Bibliography

[1] P. Lynch, J. Comput. Phys. **227**, 3431 (2008), ISSN 0021-9991, URL http://dx.doi.org/10.1016/j.jcp.2007.02.034.

[2] N. Sopipan, **8** (2014).

[3] I. Amra and A. Maghari (2018), pp. 135–140.

[4] R. Yang, L. Li, Z. Zhao, and Y. Zhang (2013), pp. 304–307.

[5] R. d. Subramanian, A. Pachiyappan, C. Venkatesh, and P. Agarwal, International Journal of Automation and Computing (2016).

[6] M. P. Darji, V. Dabhi, and H. Prajapati (2015).

[7] A. Nugroho, S. Seno Saleh, S. Hartati, and K. Mustofa, International Journal of Advanced Computer Science and Applications **5** (2014).

[8] S. Moritz and T. Bartz-Beielstein, The R Journal **9**, 207 (2017).

[9] S. A. S. Y. A. P. Maskurul Alam, Matiur Rahman, Global Journal of Science Frontier Research (2015), ISSN 2249-4626, URL https://journalofscience.org/index.php/GJSFR/article/view/1694.

[10] D. Kwiatkowski, P. C. Phillips, and P. Schmidt (1991), URL https://ideas.repec.org/p/cwl/cwldpp/979.html.

[11] S. E. Said and D. A. Dickey, Biometrika **71**, 599 (1984), ISSN 00063444, URL http://www.jstor.org/stable/2336570.

[12] R. J. Hyndman, G. Athanasopoulos, and OTexts.com, *Forecasting : principles and practice Rob J Hyndman and George Athanasopoulos* (OTexts.com [Heathmont, Victoria], 2014 2014), print edition. ed., ISBN 9780987507105.

[13] A. Trapletti and K. Hornik, *tseries: Time Series Analysis and Computational Finance* (2019), r package version 0.10-47., URL https://CRAN.R-project.org/package=tseries.

[14] P. Kapoor and S. Singh Bedi, ISRN Signal Processing **2013**, 1 (2013).

[15] R. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen, *forecast: Forecasting functions for time series and linear models* (2019), r package version 8.7, URL `http://pkg.robjhyndman.com/forecast`.

[16] R. J. Hyndman and Y. Khandakar, Journal of Statistical Software **26**, 1 (2008), URL `http://www.jstatsoft.org/article/view/v027i03`.

[17] A. Nugroho, S. Seno Saleh, S. Hartati, and K. Mustofa, International Journal of Advanced Computer Science and Applications **5** (2014).

[18] B. Pfaff, Journal of Statistical Software **27** (2008), URL `http://www.jstatsoft.org/v27/i04/`.

[19] B. Pfaff, *Analysis of Integrated and Cointegrated Time Series with R* (Springer, New York, 2008), 2nd ed., iSBN 0-387-27960-1, URL `http://www.pfaffikus.de`.

[20] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control* (Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994), 3rd ed., ISBN 0130607746.

[21] O. Karabiber and G. Xydis, Energies **12**, 928 (2019).

[22] I. Svetunkov, *greybox: Toolbox for Model Building and Forecasting* (2019), r package version 0.5.2, URL `https://CRAN.R-project.org/package=greybox`.