California State University Eastbay

Fall 2020

DELHI WEATHER FORECASTING

USING TIME SERIES FORECASTING METHODS

Submitted by

Chenwei Cao (ZJ6362)

David Liu (AZ7762)

Ankit Pushpam (QX8694)

Sayali Peshwe (WN7569)

Sruthi Bodapati (FS4443)

Supervised by

Professor Zinovy Radovilsky

SUMMARY

The weather of a place represents the state of the atmospheric environment over a brief period of time. Integrated weather conditions over several years is generally referred to as climate. Climatic change is one of the most important issues of present times. Unlike the greenhouse gases, which have a predominantly warming effect, atmospheric aerosols could either warm or cool the atmosphere depending upon the size, distribution and optical properties. Of all the climatic elements, temperature plays a major role in detecting climatic change brought about by urbanization and industrialization.

This project is undertaken to develop area specific weather forecasting models based on time series data for New Delhi, India. The main objective is to develop a univariate model for predicting the daily mean temperature. The time series data was taken out from Kaggle, whereas the data itself was originally extracted using the WUnderground API. This dataset provides data from 1st January 2013 to 24th April 2017 in the city of Delhi, India. It contains 4 features such as temperature, humidity, wind speed, and mean atmospheric pressure. The methodologies adopted to predict mean temperature were Holt-Winter's exponential smoothing model, an auto ARIMA model, a seasonal ARIMA(1,1,1)(1,1,1) model, a quadratic regression model, and a two-level quadratic regression model with residuals. Naïve forecast is used as the baseline model. We Divided the mean temperature time series into training and validation sets for model accuracy testing, i.e The dates between 2013-01-01 to 2016-12-31 is considered as the training set with 1461 training records, and the dates between 2017-01-01 to 2017-04-24 as the validation set with 114 testing records.

After rigorous evaluation, the study of our forecast models revealed that the ARIMA (1,1,1)(1,1,1) has the best model for forecasting New Delhi weather with its fitted values having the lowest RMSE (1.623) and MAPE (5.417) when compared to the Delhi mean temperature time series. Among all the models, we found that the Quadratic trend model has the highest values of RMSE (7.299) and MAPE (31.457), which is more than 5 times the error value as the ARIMA(1,1,1)(1,1,1) model. In spite of the limitations imposed by the unpredictability of the dataset, we recommend the ARIMA (1,1,1)(1,1,1) model for future mean temperature forecasts for the city of New Delhi.

INTRODUCTION

All around the world numerous weather stations are continuously monitoring the weather. And these individual weather stations collect a huge amount of information which can be used to analyse and forecast weather for future time periods. The weather stations report at regular intervals and include weather elements such as temperature, precipitations (rain, snow etc), wind, pressure and possible many more weather variables.

New Delhi, India, lies on 215m above sea level and the prevailing climate in New Delhi is known as a local steppe climate. In this project we will forecast the mean temperature of New Delhi. Usually during the year, there is little rainfall in New Delhi. And the temperature means temperature 25.2 °C. We have used the Tableau tool for Data Exploratory Analysis.

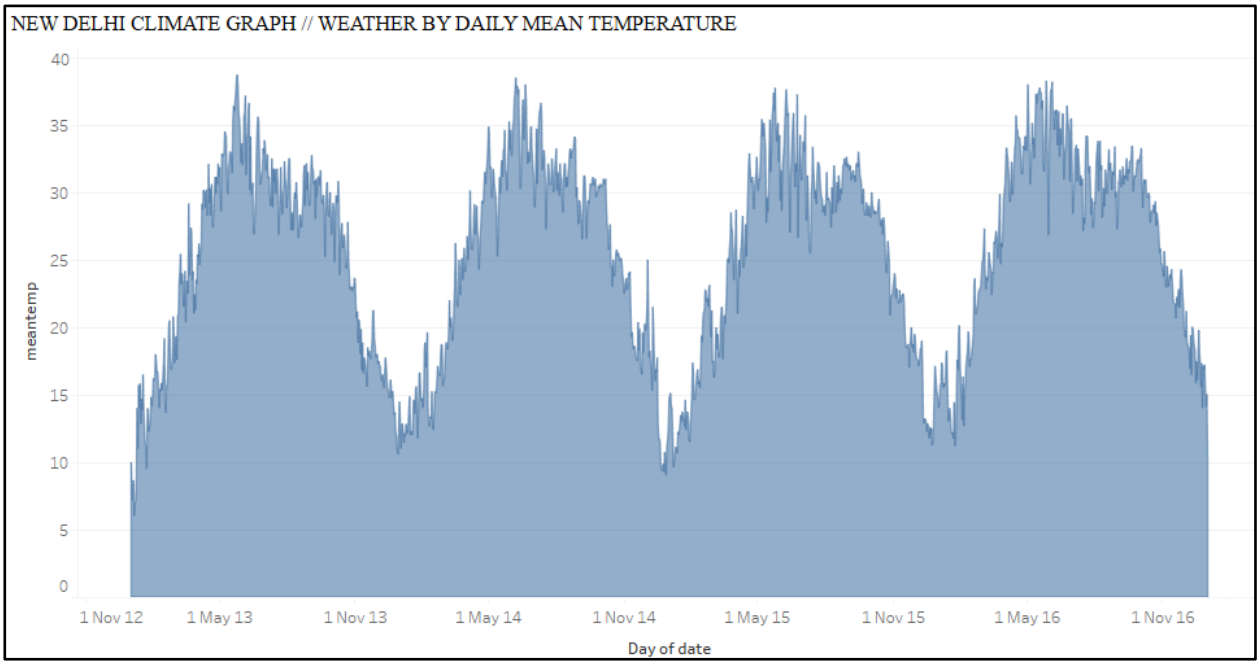

Figure 1: Daily Mean temperature for New Delhi.

In Figure 1, we see that every year, the month of June has the highest temperature, and it is also the peak summer time for that location.

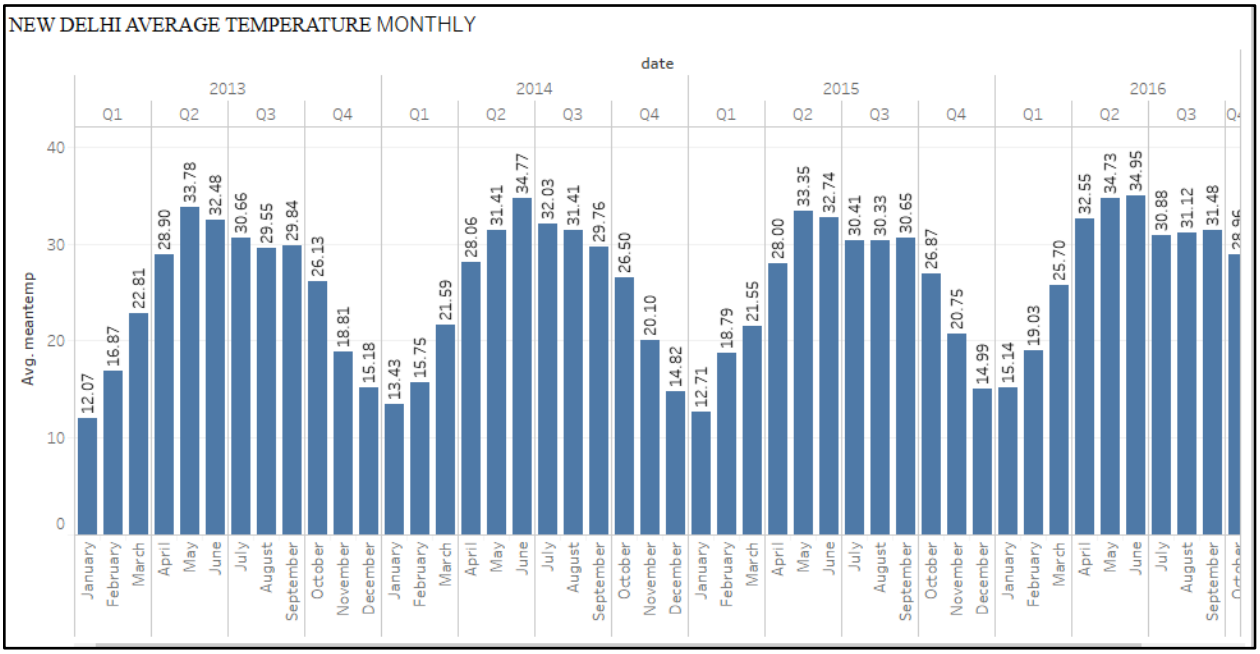

Figure 2 : Monthly average temperature.

With an average of 34.3 °C, June is the warmest month. In January, the average temperature is 14.2 °C. It is the lowest average temperature of the whole year.

One of the most important tasks is to ensure that the weather data which we have collected from the Kaggle website should be sufficient for the time series analysis. Visualization can help to determine the trend and seasonal patterns in the dataset. Figure 2 depicts delhi mean temperature variation for New Delhi.

Data set Features :

1. Date: Date of format YYYY-MM-DD
2. Meantemp: Mean temperature averaged out from multiple 3 hour intervals in a day
3. Humidity: Humidity value for the day (units are grams of water vapor per cubic meter volume of air).
4. Wind: Wind speed measured in kmph.
5. Speed: Wind speed measured in kmph.
6. Mean Pressure: Pressure reading of weather (measure in atm)

GOAL

Our group's goal for this project is to use the aforementioned daily Delhi climate data to build a predictive weather model for forecasting future daily temperatures in the city of Delhi. Our overall objective is to eventually create a univariate forecasting model that is trained with only the "meantemp" column of the dataset (daily mean temperature) which will forecast the quantitative mean centigrade temperatures for a series of days into the future. Once the model is developed, it will ideally be used for only short-term forecasts of one week into the future in order to avoid dealing with the potential inaccuracy of long term forecasts (Huijskens, 2016). Since this model will be used for weather forecasting, its usage will be daily and ongoing for as long as there are new daily temperature records observed in Delhi. Every new record will be incorporated into the model for training and for forecasting a new series of temperatures for the immediate future. In summary, this model will perform a new in-house forecast for the following week every day, as the model is retrained daily with new mean temperature records in Delhi.

To build the model, we will clean and partition the temperature data into training and validation sets. Next, we will create simple forecasting models in the R programming language, and use them to forecast the validation period. These model forecasts will be compared to the validation set and naive forecasts to measure for accuracy and performance. Lastly, all models will be used for forecasting unknown future dates, and will additionally be measured for common accuracy and performance using their fitted values. After all accuracy measurements are compared, we will then be able to select the best performing model with the lowest error rate. The following sections will discuss the model building process in further details.

DATA COLLECTION

As mentioned previously, the weather data we collected were from kaggle, and it contains a training and a validation set of weather data from Delhi that was pre-extracted using the Weather Underground API. In total, the Delhi climate dataset contains 1462 training records and 114 validation records, with a grand total of 1576 records of daily climate observations from 2013-01-01 to 2017-04-24.

```
> train.df <- read.csv("DailyDelhiClimateTrain.csv")
> test.df<-read.csv("DailyDelhiClimateTest.csv")
>
> length(train.df$date)
[1] 1462
> length(test.df$date)
[1] 114
```

Figure 3. R-code for loading the dataset.

The granularity of our dataset is set on the mean temperature of each individual day, meaning that our final model will train using daily mean temperature data, and the temperature forecasts it outputs will be the mean temperatures for each following day. This will also mean that our final weather prediction model will forecast 7 steps to represent a forecast of one week.

DATA PREPROCESS

The first step of the preprocessing phase involves checking to see whether each of the daily mean temperature records in the Delhi climate data is contiguous. To do this, the training and validation data is combined into a single 1576 record dataset. Next, a sequence of dates from 2013-01-01 to 2017-04-24 is generated into a dataframe to compare against the date period of the Delhi climate data. Shown in the results below, it is found that the generated dates only contain 1575 records, whereas the Delhi climate dataset contains 1576 records.

```
date.seq.df <- data.frame(date = c(seq(as.Date("2013-01-01"), as.Date("2017-04-24"), by="days")))

# comparing the dates in the entire climate dataset interval to the actual dates in the interval
all.equal(date.seq.df$date, combined.df$date)
```

```
> # comparing the dates in the entire climate dataset interval to the actual dates in the interval
> all.equal(date.seq.df$date, combined.df$date)
[1] "Numeric: lengths (1575, 1576) differ"
```

Figure 4

To fix this, our R code simply removes the repeated records related to a single date in the dataset, which will ensure that all rows in the dataset will only contain uniquely dated weather records. Shown in the output in the below figure, removing all records with repeated dates seem to have fixed the comparison issue. Now, the dates in the generated date set and the Delhi climate dataset are consistent and contiguous.

```
climate.df <- distinct(combined.cleaned.df, date, .keep_all = TRUE)
length(climate.df$date)

all.equal(date.seq.df$date, climate.df$date)
```

```
> all.equal(date.seq.df$date, climate.df$date)
[1] TRUE
```

Figure 5

Lastly, we will check the “meantemp” column of the dataset using the is.na() function to see if any of the values are null or missing. The codes and output below shows that there are no missing or corrupted data in the “meantemp” column. As such, no further cleaning of the dataset is necessary.

```
# check to see if any mean temperature values are null
any(is.na(climate.df$meantemp))
```

```
> any(is.na(climate.df$meantemp))
[1] FALSE
```

Figure 6

DATA VISUALIZATION AND EXPLORATION

The mean temperature data was previously visualized in the report's introduction as a visual analysis. We will further explore the data by showing its time series components. The first step is to create the time series. Shown below, the time series is a sequence of mean temperature dates with no seasonal frequency.

```
climate.ts <- ts(climate.df$meantemp, start = 1, freq = 1)
climate.ts
```

Figure 7

Time Series Component Analysis with Autocorrelation Chart

Using the generated time series, the codes below will output the autocorrelation plot with a max lag of 365 days for the mean temperature data in the Delhi climate dataset. The reason for using 365 as the max lag value is to observe the data's autocorrelation in a yearly period to check for possible patterns.

```
# checking the autocorrelation values of the mean temperature data
climate.acf <- Acf(climate.ts, lag.max = 365,
                   main = "Autocorrelation for Delhi Climate Data (Mean Temperature)")
```

Figure 8

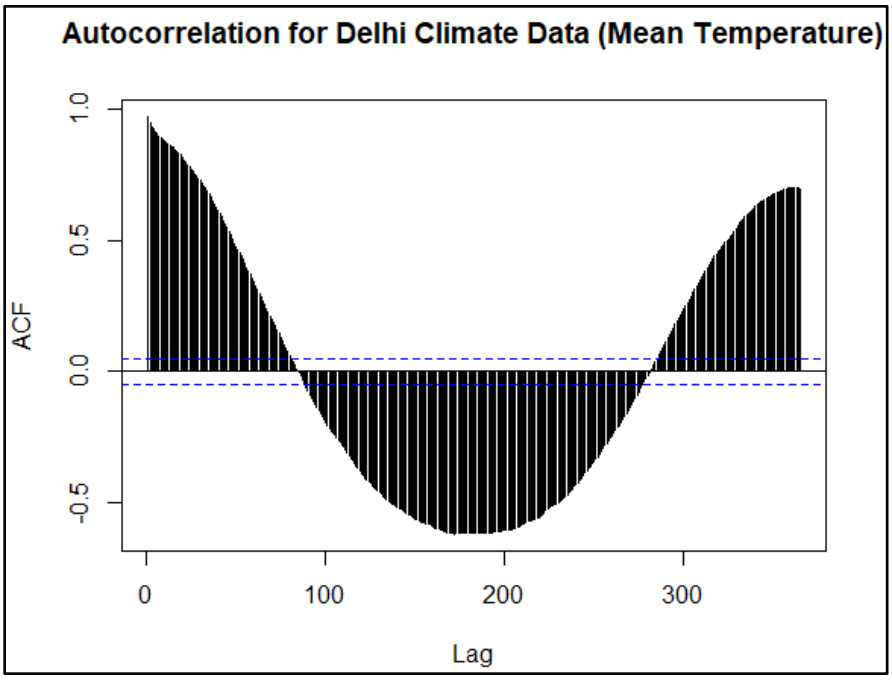

Figure 9

As shown in figure 9, the daily mean temperature data has a very high autocorrelation value for lag = 1, which suggests that the data has a strong positive trend. The high autocorrelation value at lag = 365 indicates that the data is strongly seasonal in yearly periods. Knowing this, however, the time series will not be reformatted into a 365 days seasonal period because this would not accurately indicate leap years, which are 366 days. Therefore, the time series for the mean temperatures will remain with no seasonal frequency.

Predictability Analysis using a Differenced Time Series (lag = 1) and Autocorrelation

Shown in the codes below, the Delhi weather time series is analyzed for predictability by differencing the time series with a lag of 1, and then finding its autocorrelation values with a max time lag of 365.

```
# checking the autocorrelation values of the mean temperature data with diff of lag 1
climate.ts.diff <- diff(climate.ts, lag = 1)
climate.ts.diff

climate.ts.acf = Acf(climate.ts.diff, lag.max=365,
                     main="Delhi Temperature Autocorrelation (min=1, max=365)")
climate.ts.acf
```

Figure 10

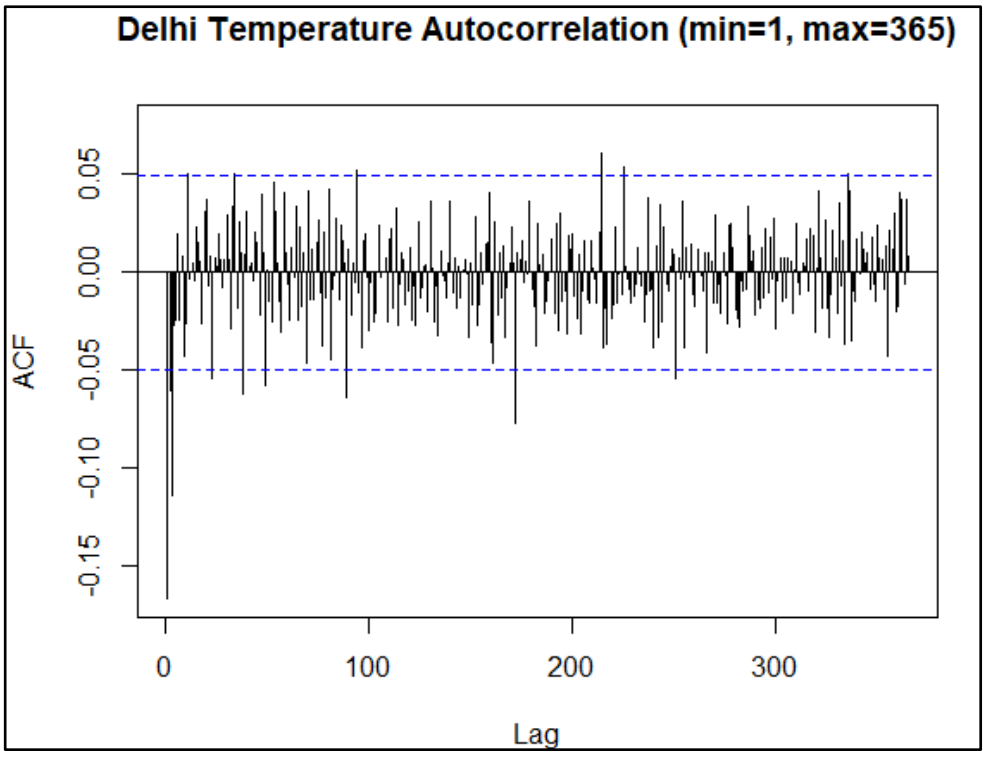

Figure 11

The above autocorrelation shows that while some of the time lag autocorrelation values are significant, the vast majority of them are not. This suggests that the mean temperature data might not be very predictable.

Predictability Analysis using First Order Arima Model

Below codes shows an ARIMA (1,0,0) model, which indicates that the model uses an autocorrelation lag that is equal to 1, but does not incorporate past forecast residuals or lag

differences. Like the differenced autocorrelation chart above, this model can also test for a dataset's predictability.

```
climate.ts.ar1 <- Arima(climate.ts, order = c(1,0,0))
summary(climate.ts.ar1)
```

```
Series: climate.ts
ARIMA(1,0,0) with non-zero mean

Coefficients:
      ar1      mean
        1    25.219
s.e.  NaN  6325.748

sigma^2 estimated as 2.84:  log likelihood=-3056.31
AIC=6118.62   AICc=6118.63   BIC=6134.71

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.01396824 1.684145 1.247856 -0.2298642 5.465201 0.9993651 -0.1662834
```

Figure 12

Running the codes creates a model with the above parameters. To assess for predictability, we look at the size of the ar1 coefficient. A larger value indicates non-predictability whereas a lower value indicates predictability. The above coefficient in the model shows a 1, which largely indicates that the Delhi mean temperature data is unpredictable.

Regardless of the unpredictable nature of the dataset, we will still create several models to test for their forecasting performance and see if it is possible to isolate a model with some capability of performing mean temperature prediction for the Delhi weather data.

TIME SERIES PARTITIONING

The mean temperature time series will be divided into a training and validation set for model accuracy testing, as well as to assess for overfitting and underfitting. Since this is a time series, the data will be partitioned, or divided, at a certain date. In this project, the dates between 2013-01-01 to 2016-12-31 will be partitioned as the training partition and the dates between 2017-01-01 to 2017-04-24 as the validation partition. The below figure shows the codes for setting the two partitions apart. Note that 114 represents the exact number of days between January 1, 2017 to April 24, 2017. In total, the partitioned data sets will have 1461 training records and 114 validation records.

```
# Number of validation days. The last 114 days of the dataset represenets the number of days from
# Jan 1 2017 to Apr 24 2017
validation <- 114
training <- length(climate.ts) - validation
train.ts <- window(climate.ts, start = 1, end = training)
valid.ts <- window(climate.ts, start = training + 1, end = training + validation)
```

Figure 13

APPLICATION OF FORECASTING METHODS

To begin the forecasting process, our project team will utilize one data-driven method and four model-based methods for prediction. Hence, the five following models were used: A Holt-Winter's Exponential Smoothing model, an auto-ARIMA model, a ARIMA(1,1,1)(1,1,1) model, a regression TSLM with quadratic trend model, and a two-level quadratic regression TSLM model with AR1 residual predictions. These models were trained using the training partition for overfitting/underfitting analysis, as well as trained using the entire dataset.

Holt-Winter's Exponential Smoothing Model

The Holt-Winter's model is trained using the training partition and the model parameters (ZZZ), as shown below. The "ZZZ" indicates that the HW model will automatically decide whether the error, trend, and seasonality parameters should be additive or multiplicative.

```
hw.zzz <- ets(train.ts, model = "zzz")
summary(hw.zzz)
```

Figure 14

```
ETS(A,N,N)

Call:
 ets(y = train.ts, model = "zzz")

  Smoothing parameters:
    alpha = 0.7808

  Initial states:
    l = 9.4362

  sigma:  1.6386

     AIC     AICc      BIC
12093.06 12093.08 12108.92

Training set error measures:
                    ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.004805309 1.637439 1.230081 -0.2892406 5.294993 0.9946848 0.03012938
```

Figure 15

Shown by the above summary, the model will have the following equation:

$F_{t+k} = L_t + kT_t + S_{t+k+M}$

Where:

$L_t = -0.7808S_{t-M} + 0.2192(L_{t-1} + T_{t-1})$

$T_t = T_{t-1}$

$S_t = S_{t-M}$

The meanings of the variables are the following:

- S_{t+k+M} is the seasonal additive component for period t + k + M
- k is the number of periods to be forecasted into the future
- t is the time period
- M is the number of seasons
- T_t is the trend at time t
- S_t is the seasonal component at time t
- L_t is the level at time t

The Holt-Winter's model was, again, trained using the entire Delhi mean temperature time series, shown in below summary.

```
ETS(A,N,N)

Call:
 ets(y = climate.ts, model = "ZZZ")

  Smoothing parameters:
    alpha = 0.7816

  Initial states:
    l = 9.4288

  sigma:  1.6555

     AIC     AICc      BIC
13187.13 13187.15 13203.22

Training set error measures:
                    ME     RMSE      MAE       MPE      MAPE      MASE      ACF1
```

Figure 16

The equation for the above summary is the following:

$F_{t+k} = L_t + kT_t + S_{t+k+M}$

Where:

$L_t = -0.7816S_{t-M} + 0.2184(L_{t-1} + T_{t-1})$

$T_t = T_{t-1}$

$S_t = S_{t-M}$

Auto-ARIMA Model

The second model is the auto-ARIMA model, which was trained with the training partition and created using the codes shown below. Ordinarily, the model requires an order and seasonal parameter, which the below model did not include. Ignoring the parameters allows the function to automatically create an optimal ARIMA model.

```
train.auto.arima <- auto.arima(train.ts)
summary(train.auto.arima)
```

```
Series: train.ts
ARIMA(2,1,2)

Coefficients:
         ar1      ar2      ma1     ma2
      1.6879  -0.6950  -1.9154  0.9220
s.e.  0.0331   0.0328   0.0191  0.0189

sigma^2 estimated as 2.522:  log likelihood=-2745.33
AIC=5500.65   AICc=5500.69   BIC=5527.08

Training set error measures:
                      ME     RMSE      MAE        MPE     MAPE      MASE         ACF1
Training set -0.001551019 1.585291 1.192886 -0.277948 5.190793 0.9646071 -0.01660322
```

Figure 17

Shown in the summary above, the auto.arima() function automatically creates a model with (2,1,2) as the order parameter. Hence, the model utilizes the following equation:

$y_t = 1.6879y_{t-1} - 0.695y_{t-2} - 1.9154e_{t-1} + 0.922e_{t-2}$

The meaning of the variables are as follows:

- t is the time period
- y_t is the prediction at time t
- $y_{t-1} \dots y_{t-p}$ is the predictor in previous lagged time periods
- $e_{t-1} \dots e_{t-2}$ is the error terms in previous lagged time periods used to identify y_t

The auto-ARIMA model was also trained using the entire Delhi mean temperature time series, shown in the summary below.

```
Series: climate.ts
ARIMA(1,0,1) with non-zero mean

Coefficients:
         ar1      ma1     mean
      0.9848  -0.2039  25.1236
s.e.  0.0046   0.0298   2.0934

sigma^2 estimated as 2.725:  log likelihood=-3024.35
AIC=6056.7   AICc=6056.73   BIC=6078.15

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.01743474 1.649219 1.24613 -0.4890669 5.480859 0.9979825 0.02156057
```

Figure 18

The equation for the above summary is as follows:

$y_t = 25.1236 + 0.9848y_{t-1} - 0.2039e_{t-1}$

ARIMA(1,1,1)(1,1,1) Model

The third model is an ARIMA(1,1,1)(1,1,1) model, which was trained using the training partition and has (1,1,1) and (1,1,1) indicated as its order and seasonal parameters, respectively.

```
seas.arima = Arima(train.ts, order = c(1,1,1), seasonal = c(1,1,1))
summary(seas.arima)
```

```
Series: train.ts
ARIMA(1,1,1)

Coefficients:
         ar1      ma1
      0.5738  -0.8044
s.e.  0.0414   0.0285

sigma^2 estimated as 2.576:  log likelihood=-2761.42
AIC=5528.83   AICc=5528.85   BIC=5544.69

Training set error measures:
                      ME     RMSE      MAE        MPE     MAPE      MASE         ACF1
Training set 0.009042973 1.603248 1.216179 -0.3415995 5.274075 0.9834428 0.009074391
```

Figure 19

The above summary shows that the created model no longer contains the seasonal parameter of (1,1,1) because the Delhi mean temperature time series does not have seasonality. As a result, the model contains the following equation:

$y_t = 0.5738y_{t-1} - 0.8044e_{t-1}$

The meaning of the variables are as follow:

- t is the time period
- y_t is the prediction at time period t
- $y_{t-1} \dots y_{t-p}$ is the predictor in previous lagged time periods
- $e_{t-1} \dots e_{t-2}$ is the error terms in previous lagged time periods used to identify y_t

The ARMIA(1,1,1)(1,1,1) model was also trained using the entire Delhi mean temperature time series, shown in the summary below.

```
Series: climate.ts
ARIMA(1,1,1)

Coefficients:
          ar1      ma1
      0.5794  -0.8041
s.e.  0.0410   0.0285

sigma^2 estimated as 2.641:  log likelihood=-2996.73
AIC=5999.47   AICc=5999.48   BIC=6015.55

Training set error measures:
                     ME     RMSE      MAE         MPE    MAPE      MASE          ACF1
Training set 0.03106523 1.623491 1.230415 -0.2754411 5.41659 0.9853965 0.0009038344
```

Figure 20

The equation for the above summary is as follow:

$y_t = 0.5794y_{t-1} - 0.8041e_{t-1}$

TSLM Model with Quadratic Trend

The TSLM regression model with quadratic trend was trained with the training partition and created using the codes shown below. The trend + I(trend^2) parameter gives an indication that the model is quadratic.

```
TSLM.quad  <- tslm(train.ts ~ trend + I(trend^2))
summary(TSLM.quad)
```

```
Call:
tslm(formula = train.ts ~ trend + I(trend^2))

Residuals:
    Min      1Q  Median      3Q     Max
-17.707  -6.446   2.110   5.741  14.622

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.369e+01  5.721e-01  41.417   <2e-16 ***
trend        2.803e-03  1.807e-03   1.551    0.121
I(trend^2)  -3.310e-07  1.197e-06  -0.277    0.782
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.279 on 1458 degrees of freedom
Multiple R-squared:  0.01782,   Adjusted R-squared:  0.01647
F-statistic: 13.23 on 2 and 1458 DF,  p-value: 2.031e-06
```

Figure 21

It would appear that neither of the linear or quadratic trend coefficients are statistically significant. In addition, the multiple r-squared and adjusted r-squared values are very low. However, the p-value is quite low and significant. The above summary indicates that the model equation is the following:

$y_t = 23.69 + 0.002803t - 0.000000331t^2$

The meaning of the variables are as follow:

- y_t is the prediction at time period t
- t is the time period

The LSTM model with quadratic trends was also trained using the entire Delhi mean temperature time series, shown in below summary.

```
Call:
tslm(formula = climate.ts ~ trend + I(trend^2))

Residuals:
    Min      1Q  Median      3Q     Max
-17.040  -6.531   1.884   5.965  14.828

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.301e+01  5.530e-01  41.607  < 2e-16 ***
trend        6.581e-03  1.621e-03   4.061 5.12e-05 ***
I(trend^2)  -3.583e-06  9.957e-07  -3.599 0.000329 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.306 on 1572 degrees of freedom
Multiple R-squared:  0.01149,   Adjusted R-squared:  0.01023
F-statistic: 9.136 on 2 and 1572 DF,  p-value: 0.0001135
```

Figure 22

The equation for the above summary is as follow:

$y_t = 23.01 + 0.006581t - 0.000003583t^2$

TSLM Model with Quadratic Trend + AR1 Residuals

The final model, a TSLM Model with Quadratic Trend and AR1 Residuals, is a simple combination of the TSLM model with quadratic trends and a first order ARIMA model that is trained with the TSLM model's residuals. The codes for creating the residual model is as follow:

```
TSLM.quad.pred.res.ar1 <- Arima(TSLM.quad.pred$residuals, order=c(1,0,0))
summary(TSLM.quad.pred.res.ar1)
```

```
Series: TSLM.quad.pred$residuals
ARIMA(1,0,0) with non-zero mean

Coefficients:
         ar1     mean
      0.9751  -0.6222
s.e.  0.0059   1.7003

sigma^2 estimated as 2.747:  log likelihood=-2811.78
AIC=5629.56   AICc=5629.57   BIC=5645.42

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE     MASE        ACF1
Training set 0.01484953 1.656304 1.24232 -17.24969 88.17933 1.004603 -0.1441893
```

Figure 23

Above summary indicates that the equation for the residual AR1 model is as follow:

$y_t = -0.6222 + 0.9751y_{t-1}$

The meaning of the variables are as follow:

- t is the time period
- y_t is the prediction at time period t
- $y_{t-1} \ldots y_{t-p}$ is the predictor in previous lagged time periods

As a result, the final equation for the two-level model would be equivalent to the following:

$y_t = (-0.6222 + 0.9751y_{t-1}) + (23.69 + 0.002803t - 0.000000331t^2)$

The two-level model was also trained using the entire Delhi mean temperature time series. The AR1 of the quadratic LSTM model's residuals is shown in the summary below.

```
Series: TSLM.quad.entire.pred$residuals
ARIMA(1,0,0) with non-zero mean

Coefficients:
      ar1     mean
        1  -0.0042
s.e.  NaN      NaN

sigma^2 estimated as 2.84:  log likelihood=-3056.29
AIC=6118.58   AICc=6118.59   BIC=6134.66

Training set error measures:
                      ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
Training set 0.01303468 1.684123 1.247838 56.20181 165.1909 0.9993653 -0.1663064
```

Figure 24

The equation for the above summary is as follow:

$y_t = -0.0042 + y_{t-1}$

Altogether, the two-level model equation for the entire Delhi climate dataset is the following:

$y_t = (y_t = -0.0042 + y_{t-1}) + (23.01 + 0.006581t - 0.000003583t^2)$

Naive Model

This model is only used for assessing the above model's performance. The codes below create a naive model for both the training and entire data set.

```
naive.temp <- naive(train.ts, h=114)
naive.temp

naive.entire.temp <- naive(climate.ts, h=7)
naive.entire.temp
```

Figure 25

These naive() functions will create (h) number of naive forecasts into the future. These forecasts are the actual values in the time series moved forward in time by h number of steps.

MODEL EVALUATION

All models which were trained with the training data partition were used to forecast ahead 114 days into the validation period. These forecasted mean temperatures were then compared to the validation values to assess for accuracy.

Below table compares the accuracies for each of the models when forecasting into the validation period. Only the RMSE & MAPE metrics were considered for performance evaluation:

	Models	RMSE	MAPE
1	HW Model	9.299	28.510
2	Auto Arima Model	12.255	40.656
3	Arima(1,1,1)(1,1,1)	8.802	26.482
4	Quad TSLM	8.417	42.743
5	Two Level	4.684	21.695
6	Naive	9.202	28.085

Figure 26

All models which were trained with the entire Delhi mean temperature time series were used to forecast ahead 7 days into the future. The fitted values for these models were then compared to the entire mean temperature time series.

Below table compares the common accuracies for each of the models' fitted values when compared to the entire mean temperature series. Only the RMSE & MAPE metrics were considered for performance evaluation:

	Models.entire	RMSE.entire	MAPE.entire
1	HW Model	1.654	5.437
2	Auto Arima Model	1.649	5.481
3	Arima(1,1,1)(1,1,1)	1.623	5.417
4	Quad TSLM	7.299	31.457
5	Two Level	1.684	5.465
6	Naive	1.685	5.469

Figure 27

CONCLUSION AND IMPLEMENTED MODEL

Accuracy Measures help us assess the model performance. Following are the accuracy measures that help identify models with best predictive accuracy.

RMSE: Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors).

MAE: Mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon.

MPE: In statistics, the mean percentage error (MPE) is the average computed percentage of errors by which forecasts differ.

MAPE: It measures the overall accuracy as a percentage, and it can be evaluated as the average absolute percent error for each time-period divided by actual values.

We will consider RMSE and MAPE as a metric for comparing the models.

Our preliminary evaluation of model performance is based on calculating the accuracy measures for the validation data partition. We report the lowest RMSE (4.684) and MAPE (21.695) values for the two-level model (with quadratic trend and seasonality and AR(1) model for residuals). The next better model is ARIMA (1,1,1)(1,1,1) with calculated RMSE (8.802) and MAPE(26.482). All models, except for the MAPE comparison for the auto-ARIMA model, performed better than the naive accuracy.

Prior to attempting to forecast the future periods of the time series, all the data partitions (the training and validation periods) are recombined into one entire time series dataset. Applying these models to the entire data set, we further evaluate the model performance based on the accuracy measures. Naïve Forecast is used as the baseline model, which provides an RMSE (9.202)

and MAPE (28.085) for the entire dataset. Based on the RMSE and MAPE measures, ARIMA (1,1,1)(1,1,1) model for forecasting has the lowest RMSE (1.623) and MAPE (5.417) as compared to all other models considered in this particular analysis. The Quadratic trend model for forecasting has the highest values of RMSE (7.299) and MAPE (31.457) (more than 5 times than the ARIMA(1,1,1)(1,1,1) model) among all the models and hence is the least accurate model for predicting future periods.

The two level model (Quadratic trend and seasonality with AR(1) model for residuals) was the most accurate model for the validation partition, but ARIMA (1,1,1)(1,1,1) turned out to be the most accurate for the entire dataset, suggesting that the two-level model overfitted with the data. Additionally, all models, excluding the auto-ARIMA and Quadratic TSLM model, performed better than the naive model. Therefore, we recommend the ARIMA(1,1,1)(1,1,1) model as our most accurate model to use for forecasting Delhi weather (Temperatures). As for the dataset considered in our analysis, the predictability of the data is not very good which can be inferred from the autocorrelation plot of the differenced series, as well as the AR1 model.

One of the major benefits of Time Series analysis is that it enables analysts to forecast for future time periods. Time Series analysis helps in discovering data patterns and seasonality in the past data and in turn, facilitates to gain insights from the historical data. Meaningful insights derived from the available historical data can be utilized to predict future values. Time Series forecasting is one of the most powerful components in Business Analysis. Forecasting ensures optimal business decisions and reforms.

In contrast to these very powerful applications and benefits, forecasts are never fully (100%) accurate. A single chance event in the data can affect all future forecasts and may render erroneous results. Advanced time series models and applications can be highly time consuming

and also involve higher costs. Huge amount of time is invested in data collection and organization sometimes. These limitations, well, can be overcome by identifying and selecting most optimal methods and tools.

BIBLIOGRAPHY

Huijskens, Thomas. “Long-Term Forecasting with Machine Learning Models.” 3 Aug. 2016, https://thuijskens.github.io/2016/08/03/time-series-forecasting/.

APPENDICES

Dataset : https://www.kaggle.com/sumanthvrao/daily-climate-time-series-data

Snapshot of the training data partition “train.ts”:

```
> train.ts
Time Series:
Start = 1
End = 1461
Frequency = 1
   [1] 10.000000  7.400000  7.166667  8.666667  6.000000  7.000000  7.000000  8.857143 14.000000 11.000000
  [11] 15.714286 14.000000 15.833333 12.833333 14.714286 13.833333 16.500000 13.833333 12.500000 11.285714
  [21] 11.200000  9.500000 14.000000 13.833333 12.250000 12.666667 12.857143 14.833333 14.125000 14.714286
  [31] 16.200000 16.000000 16.285714 18.000000 17.428571 16.625000 16.666667 15.600000 14.000000 15.428571
  [41] 15.250000 15.875000 15.333333 16.285714 17.333333 19.166667 14.428571 13.666667 15.600000 15.857143
  [51] 17.714286 20.000000 20.500000 17.428571 16.857143 16.875000 17.857143 20.800000 19.428571 17.333333
  [61] 19.000000 19.333333 17.600000 20.875000 20.857143 23.428571 24.166667 25.428571 23.142857 24.000000
  [71] 23.500000 21.500000 22.333333 24.166667 20.333333 22.666667 23.428571 22.500000 29.166667 23.833333
  [81] 25.250000 27.375000 27.000000 23.500000 24.142857 21.000000 22.428571 21.250000 23.500000 23.200000
  [91] 25.375000 25.166667 26.200000 24.600000 25.600000 25.857143 29.142857 28.714286 30.166667 30.000000
 [101] 30.000000 28.857143 30.200000 28.250000 28.250000 32.125000 29.200000 30.285714 28.285714 30.625000
 [111] 27.666667 27.375000 28.625000 30.285714 31.142857 29.875000 31.142857 30.571429 32.125000 31.142857
 [121] 31.857143 29.833333 28.571429 32.857143 32.625000 32.750000 32.875000 34.500000 34.285714 34.000000
 [131] 30.750000 29.857143 31.714286 32.285714 33.000000 33.000000 32.833333 31.400000 35.333333 36.400000
```

Figure 27

Snapshot of the validation data partition “valid.ts”:

```
> valid.ts
Time Series:
Start = 1462
End = 1575
Frequency = 1
  [1] 10.00000 18.50000 17.11111 18.70000 18.38889 19.31818 14.70833 15.68421 14.57143 12.11111 11.00000
 [12] 11.78947 13.23529 13.20000 16.43478 14.65000 11.72222 13.04167 14.61905 15.26316 15.39130 18.44000
 [23] 18.11765 18.34783 21.00000 16.17857 16.50000 14.86364 15.66667 16.44444 16.12500 15.25000 17.09091
 [34] 15.63636 18.70000 18.63158 16.88889 15.12500 15.70000 15.37500 14.66667 15.62500 16.25000 16.33333
 [45] 16.87500 17.57143 20.25000 21.30000 21.12500 22.36364 23.37500 21.83333 19.12500 18.62500 19.12500
 [56] 19.00000 18.75000 19.87500 23.33333 24.46154 23.75000 20.50000 19.12500 19.75000 20.00000 22.62500
 [67] 21.54545 20.78571 19.93750 18.53333 17.37500 17.44444 18.00000 19.87500 24.00000 20.90000 24.69231
 [78] 24.66667 23.33333 25.00000 27.25000 28.00000 28.91667 26.50000 29.10000 29.50000 29.88889 31.00000
 [89] 29.28571 30.62500 31.37500 29.75000 30.50000 30.93333 29.23077 31.22222 27.00000 25.62500 27.12500
[100] 27.85714 29.25000 29.25000 29.66667 30.50000 31.22222 31.00000 32.55556 34.00000 33.50000 34.50000
[111] 34.25000 32.90000 32.87500 32.00000
```

Figure 28