DELHI WEATHER PREDICTION

Time Series Term Project

Udbhav Kush
ukush4@gwu.edu

Table of Contents

TABLE OF FIGURE, TABLES, AND EQUATIONS

ABSTRACT

This study focuses on the application of time series analysis to predict the daily weather conditions in Delhi. By analyzing historical weather data, including variables such as temperature, humidity, precipitation, and wind speed, a robust predictive model is developed. The study aims to provide accurate and reliable forecasts, enabling individuals, industries, and authorities to make informed decisions based on anticipated weather patterns. The results highlight the efficacy of time series analysis in predicting daily weather, contributing to improved planning, risk management, and resource allocation in Delhi. The findings have implications for various sectors, including agriculture, transportation, and urban infrastructure, fostering resilience and sustainability in the face of changing weather dynamics.

INTRODUCTION

This report presents an in-depth analysis of the Delhi weather time series data, aiming to develop accurate forecasting models and gain insights into the daily weather patterns. The analysis encompasses essential steps such as dataset cleaning, outlier removal, stationarity checks, order determination for the ARIMA process, performance comparison of various models, and the incorporation of Long Short-Term Memory (LSTM) as a deep learning method. The objective is to identify the most effective model for predicting Delhi weather.

The initial phase of the analysis involves cleaning the dataset to ensure data integrity by addressing missing values, inconsistencies, and erroneous entries. Outliers, if present, will be identified and removed to mitigate their impact on the subsequent modeling process, as they can distort the statistical properties of the time series.

To incorporate linear time series models, a stationarity check will be conducted. Stationarity, an essential assumption for many modeling techniques, will be assessed using appropriate methods. If required, suitable transformations will be applied to achieve stationarity.

The order of the ARIMA process will be determined using techniques such as Autocorrelation Function (ACF), Partial Autocorrelation Function (PACF), and Generalized Partial Autocorrelation (GPAC). These techniques aid in identifying the optimal lag structure and parameters, improving the predictive capabilities of the ARIMA model.

In addition to linear models, various base models will be employed to capture the complexities and dynamics of the time series. These models will be trained and evaluated using appropriate performance metrics to compare their forecasting accuracy. Furthermore, LSTM, a powerful deep learning algorithm, will be incorporated to model the time series. Through performance comparison, the most effective approach for predicting the Delhi weather will be determined.

This comprehensive analysis aims to provide valuable insights into the daily weather patterns in Delhi and identify the most accurate forecasting model. The findings will contribute to improved understanding, planning, and decision-making in sectors influenced by weather dynamics, such as agriculture, transportation, and urban infrastructure.

DESCRIPTION OF DATASET

Initial data was sampled hourly and had a lot of missing values. To address the issue of numerous missing values in the original hourly dataset, a decision was made to resample the data to a daily frequency. By transitioning to a daily dataset, we ensured a more comprehensive and reliable dataset for analysis, free from the gaps caused by missing values.

- Dependent Variable: In this analysis, the dependent variable chosen for the study is the temperature, represented as 'temp' in the dataset. The unit of temperature used in this analysis is degree Celsius.
- Independent Variables:
    - windgust: wind gust is the sudden increase in speed of wind in kph.
    - vis is visibility (in kms)
    - dewpoint in degree Celsius: atmospheric temperature at which water droplets begins to condense.
    - snow in inches: the height of snow fall.
    - windspeed in kph: the speed of wind.
    - wdird is wind direction in degrees.
    - pressure is in millibars (mb): atmospheric pressure.
    - humidity is in percentage.
    - windchill in kph.
    - heatindex in degree Celsius.
    - Precipitation in mm.

- Preprocessing Dataset

#	Column	Non-Null Count	Dtype
0	Datetime	100990 non-null	object
1	conditions	100918 non-null	object
2	dewpoint	100369 non-null	float64
3	fog	100990 non-null	int64
4	hail	100990 non-null	int64
5	heatindex	29155 non-null	float64
6	humidity	100233 non-null	float64
7	precipitation	0 non-null	float64
8	pressure	100758 non-null	float64
9	rain	100990 non-null	int64
10	snow	100990 non-null	int64
11	temp	100317 non-null	float64
12	thunder	100990 non-null	int64
13	tornado	100990 non-null	int64
14	visibility	96562 non-null	float64
15	wdirdegrees	86235 non-null	float64
16	winddirection	86235 non-null	object
17	windgust	1072 non-null	float64
18	windchill	579 non-null	float64
19	windspeed	98632 non-null	float64

dtypes: float64(11), int64(6), object(3)

Table 1: Information of the Original Dataset

From the screenshot above, we can see that there are very less values of heatindex, humidity, precipitation, windgust, and windchill. Imputing these columns will not be a good idea as a lot of generated values will affect the authenticity of data. Hence, dropping these columns.

After dropping these columns, the dataset was resampled to a daily dataset instead of hourly dataset to get rid of the missing values in the other columns.

Two new columns were added to the dataset called minTemp and maxTemp indicating the minimum temperature of the day and maximum temperature of the day. This was done to get a better understanding of the weather change in a day.

The screenshot after doing all the preprocessing mentioned above is following:

```
#   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   dewpoint     7476 non-null   float64
 1   fog          7476 non-null   float64
 2   hail         7476 non-null   float64
 3   humidity     7476 non-null   float64
 4   pressure     7476 non-null   float64
 5   rain         7476 non-null   float64
 6   snow         7476 non-null   float64
 7   temp         7476 non-null   float64
 8   thunder      7476 non-null   float64
 9   tornado      7476 non-null   float64
 10  visibility   7476 non-null   float64
 11  wdirdegrees  7476 non-null   float64
 12  windspeed    7476 non-null   float64
 13  minTemp      7476 non-null   float64
 14  maxTemp      7476 non-null   float64
dtypes: float64(15)
```

Table 2: Information of the processed Dataset

From the screenshot above, we can see that we have a total of 7476 values in our daily dataset now and none of the columns have missing values.

Head of the dataset is as follows:

	dewpoint	fog	hail	humidity	pressure	rain	snow	temp	thunder	tornado	visibility	wdirdegrees	windspeed	minTemp	maxTemp
Datetime															
1996-11-01	11.666667	0.0	0.0	52.916667	-2659.666667	0.0	0.0	22.333333	0.0	0.0	2.250000	23.333333	2.466667	19.0	30.0
1996-11-02	10.458333	0.0	0.0	48.625000	1009.833333	0.0	0.0	22.916667	0.0	0.0	3.476190	106.666667	8.028571	17.0	31.0
1996-11-03	12.041667	0.0	0.0	55.958333	1010.500000	0.0	0.0	21.791667	0.0	0.0	2.286364	106.666667	4.804545	16.0	29.0
1996-11-04	10.222222	0.0	0.0	48.055556	1011.333333	0.0	0.0	22.722222	0.0	0.0	2.326667	55.555556	1.964706	15.0	29.0
1996-11-05	8.200000	0.0	0.0	29.400000	1011.800000	0.0	0.0	27.800000	0.0	0.0	3.900000	208.000000	10.020000	24.0	30.0

Table 3: First Five Values of the Dataset

In the dataset, we have a few outliers for the temperature column.

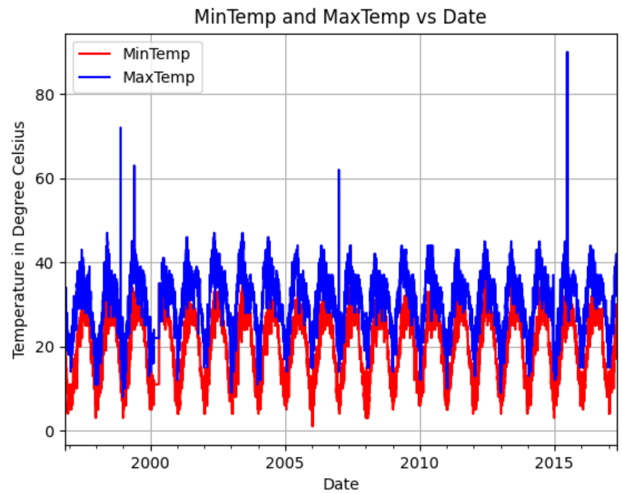


Figure 1: Minimum and Maximum Temperature VS Date Before Removing Outliers

As seen from the plot above, we can see that some temperature values are above 50 degrees Celsius. This is not possible as the highest temperature recorded in New Delhi is 48.4 degrees Celsius (reference: Wikipedia page). So, clearly these are outliers and are wrong values in the dataset. Hence, removing these outliers.

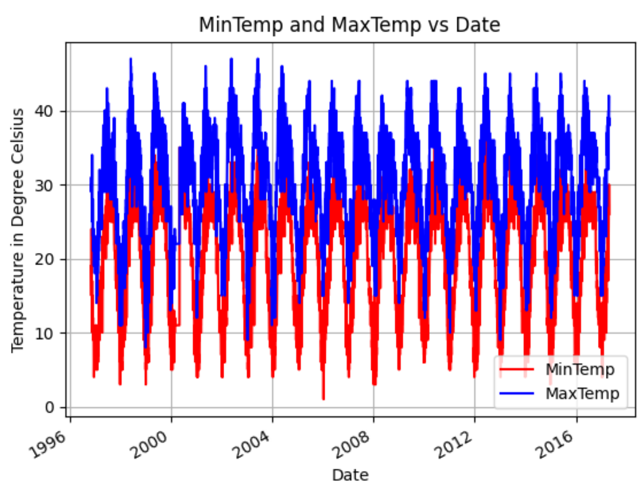


Figure 2: Minimum and Maximum Temperature VS Data After Removing Outliers

Plot of Dependent Variable VS Time

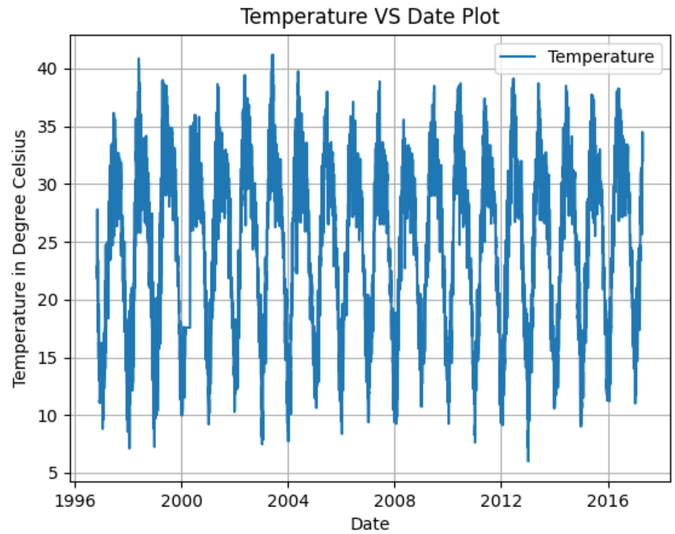


Figure 3: Dependent Variable VS Time

From the plot above, we can see that the temperature in Delhi follows a seasonal pattern. There are some clear top peaks and down peaks indicating the seasonality of the data. And, overall the range of temperature variation in Delhi is between 5 Degree Celsius and 45 degrees Celsius.

ACF/PACF of the Dependent Variable

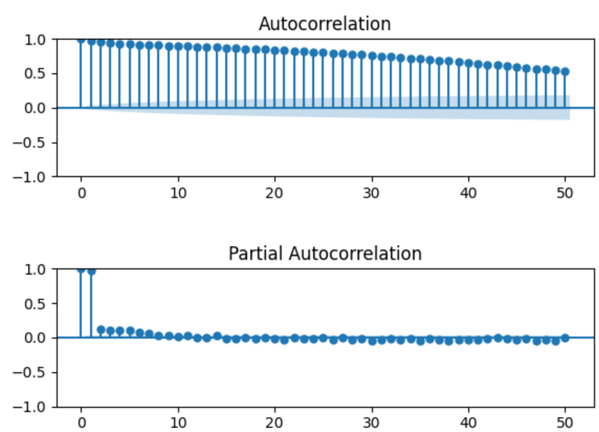


Figure 4: ACF/PACF of the Dependent Variable

From the ACF plot above, we can see that the ACF values are tailing off as the number of lags are increasing. PACF plot suggests that there is cutoff observed at the first lag. The ACF and PACF plot suggests that the process we have here is an AR process with order of 1 i.e., ARMA (1, 0). We will get more insights on the order of ARMA process in the later section of the report.

But when we increase the number of lags for the ACF plot, following is the plot:

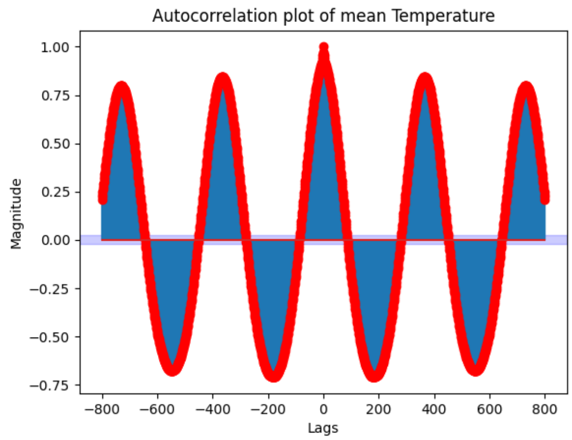


Figure 5: ACF Plot with 800 Lags

From the ACF plot above, we can see that there is a very high seasonality. It is observed from the plot that the peak occurs at the lag value 365. This indicates that the dataset is highly seasonal with a seasonality index of 365.

Correlation Matrix with seaborn heatmap

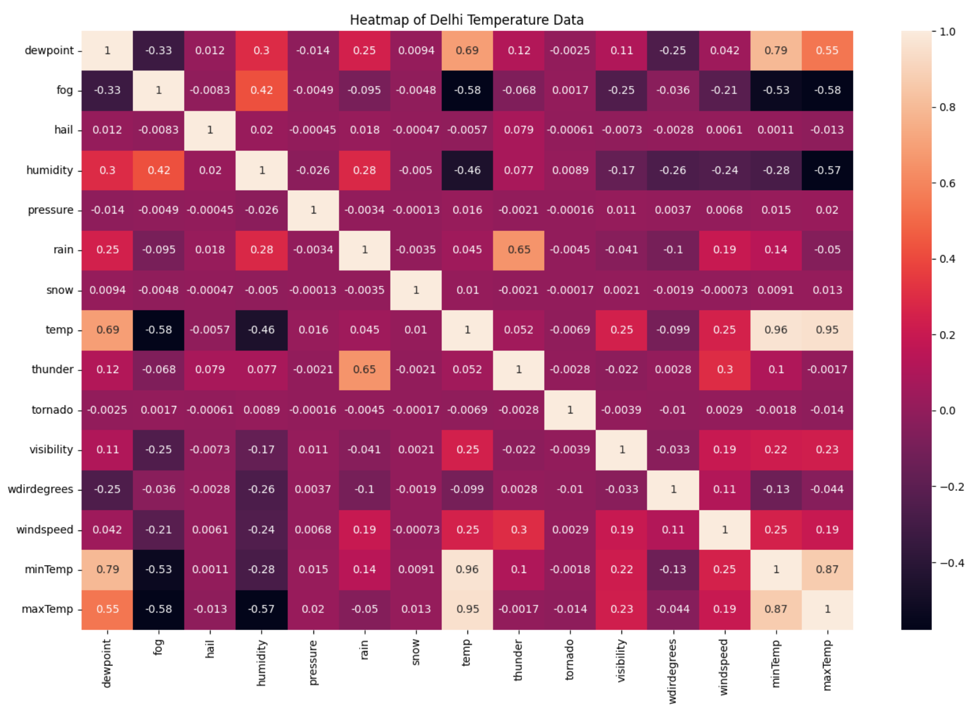


Figure 6: Correlation Matrix Heatmap

From the correlation matrix above, we can see that our dependent variable ‘temp’ is linearly correlated with ‘humidity’, ‘fog’, and ‘dewpoint’. There is a negative correlation with ‘fog’ and ‘humidity’ and positive correlation with ‘dewpoint’ with ‘temp’ (dependent variable for our dataset).

Splitting the Dataset

```
Length of X_train and y_train: 5980
Length of X_test and y_test: 1496
```

Figure 7: Splitting the Dataset

STATIONARITY

Rolling Mean and Variance Plot

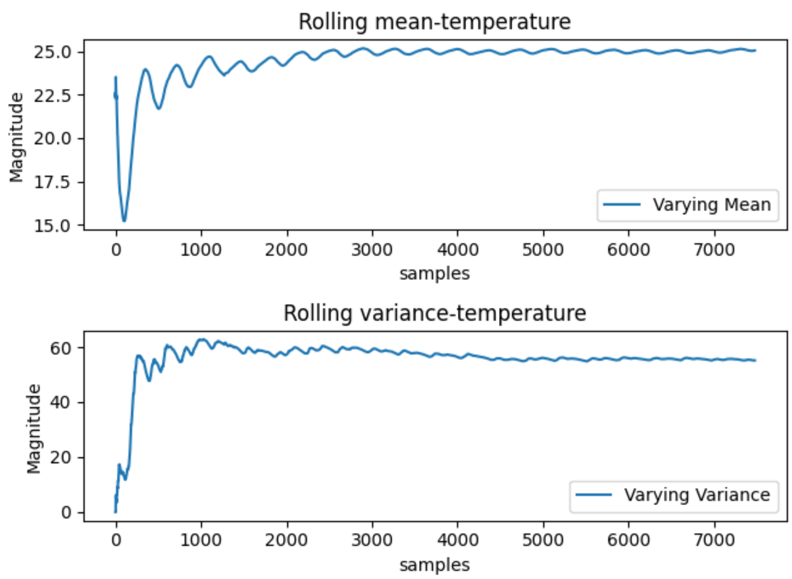


Figure 8: Rolling Mean and Variance

From the rolling mean plot, we can see that the mean is stabilizing after a few initial samples and is constant thereafter.

From the rolling variance plot, we can see that the variance is stabilizing after a few initial samples and constant thereafter.

Both these plots imply that the time series is stationary but to confirm our findings of the plot, we need to do ADF test and KPSS test.

Augmented Dickey–Fuller Test (ADF)

```
ADF TEST
NULL Hypothesis: Unit root is present i.e., time series is not stationary.
ALTERNATE Hypothesis: unit root is not present i.e., time series is stationary.
ADF Statistic: -7.178349
p-value: 0.000000
Critical Values:
    1%: -3.431
    5%: -2.862
    10%: -2.567
Rejecting the NULL hypothesis with more than 95% confidence interval
Time series is stationary
```

Figure 9: ADF Test Results

NULL Hypothesis: Unit root is present i.e., time series is not stationary.

ALTERNATE Hypothesis: unit root is not present i.e., time series is stationary.

From the screenshot above, we can see that p-value is zero indicating that we can reject the NULL hypothesis with more than 95% confidence interval and hence time series is stationary.

Kwiatkowski–Phillips–Schmidt–Shin Test (KPSS)

```
KPSS TEST
NULL Hypothesis: Time series is stationary.
ALTERNATE Hypothesis: Time series is not stationary.
Results of KPSS Test:
Test Statistic             0.032249
p-value                    0.100000
Lags Used                 53.000000
Critical Value (10%)       0.347000
Critical Value (5%)        0.463000
Critical Value (2.5%)      0.574000
Critical Value (1%)        0.739000
dtype: float64
Cannot reject the NULL hypothesis with 95% confidence interval
Time series is stationary
```

Figure 10: KPSS Test Results

NULL Hypothesis: Time series is stationary.

ALTERNATE Hypothesis: Time series is not stationary.

From the screenshot above, we can see that we have a high p-value of the test Statistic. It means that we cannot reject the NULL hypothesis with confidence and hence time series is stationary.

TIME SERIES DECOMPOSITION

Seasonally Adjusted Plot

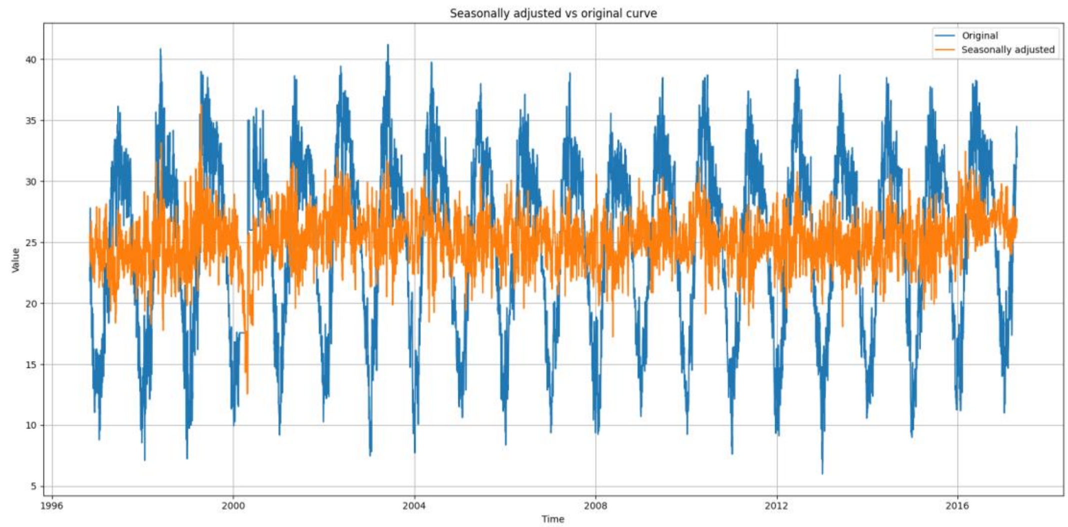


Figure 11: Seasonally Adjusted Plot VS Original

The plot suggests that the time series have high seasonality as after removing the seasonal component of the time series, there is a huge difference.

Detrended Plot

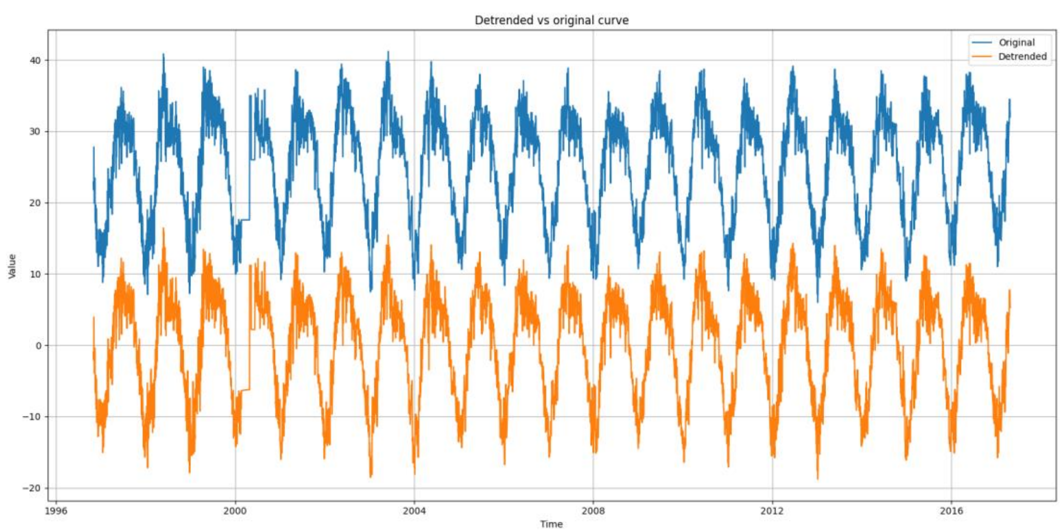


Figure 12: Detrended Plot VS Original

The original and the detrended plot seems the same with just amplitude of the time series getting affected. Everything else seems similar indicating that the time series does not have much trend component.

Strength of Trend and Seasonality

```
Strength of Trend for this dataset is  0.15689768332064336
Strength of seasonality for this dataset is  0.9318827250159856
```

Figure 13: Strength of Trend and Seasonality

From the screenshot above, we can see that the dataset has a high seasonal component as strength of trend came out to be around 93% which makes sense as we got similar results from the ACF plot.

Rubric to measure strength of Trend is given as:

$$F_T = \max\{0, 1 - \frac{Var(R_t)}{Var(T_t + R_t)}\}$$

Equation 1: Strength of Trend Rubric

Rubric to measure strength of Seasonality is given as:

$$F_S = \max\{0, 1 - \frac{Var(R_t)}{Var(S_t + R_t)}\}$$

Equation 2: Strength of Seasonality Rubric

HOLT-WINTERS METHOD

The Holt-Winters method package was applied to the train set and then the model was used to predict on the test set.

The following results were obtained:

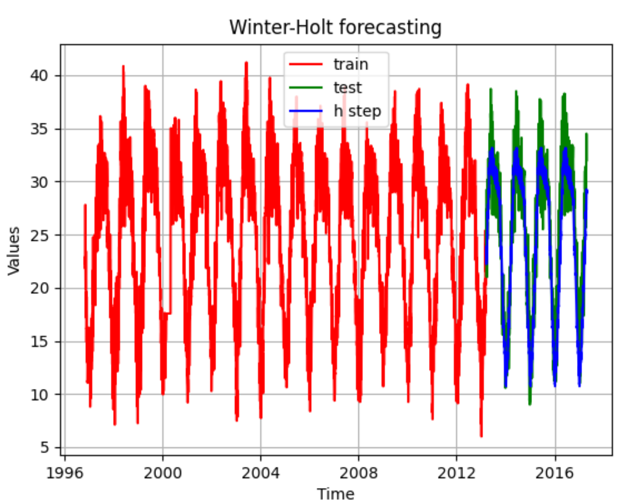


Figure 14: Holt-Winters Method Prediction Curve

From the plot above, we can see that the Holt-Winters method is predicting very well on the project's time series dataset. This forecasting technique is capturing the seasonal component very well as observed from the plot.

```
MSE for Winter-Holt method: 7.63
RMSE for Winter-Holt method: 2.76
MAE for Holt-Winter method: 2.19
```

Figure 15: Performance Metrics

From the performance metrics figure, we can see that the Winter-Holt method is doing a good job in forecasting the test set values.

FEATURE SELECTION/ELIMINATION AND MULTIPLE LINEAR REGRESSION

Checking Collinearity

```
condition_number: 2.1862590788520824e+16
SingularValues = [6.04547963e+13 2.70082940e+08 6.71739117e+06 3.01889313e+05
 1.55463683e+05 2.52453074e+04 1.05635899e+02 7.93690029e+01
 1.54137793e+01 1.27132273e-01 1.81119768e-02 2.11179730e-15]
```

Figure 16: Collinearity Check using Condition Number and SVD Analysis

From the screenshot above, we can see that there is a high collinearity in the dataset. Condition number is of the order of 10^{16} which is a very high number indicating very high collinearity.

The singular value analysis also suggests that there is high collinearity as the starting values in the array are very high indicating high collinearity.

Backward Stepwise Regression

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   temp   R-squared:                       0.955
Model:                            OLS   Adj. R-squared:                  0.955
Method:                 Least Squares   F-statistic:                 1.141e+04
Date:                Wed, 10 May 2023   Prob (F-statistic):               0.00
Time:                        18:06:44   Log-Likelihood:                -11278.
No. Observations:                5980   AIC:                         2.258e+04
Df Residuals:                    5968   BIC:                         2.266e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
constant      24.8807      0.021   1204.867      0.000      24.840      24.921
dewpoint       6.9032      0.026    263.669      0.000       6.852       6.954
fog            0.4293      0.027     15.760      0.000       0.376       0.483
hail           0.0406      0.021      1.955      0.051      -0.000       0.081
humidity      -5.8197      0.029   -200.800      0.000      -5.877      -5.763
pressure       0.0783      0.021      3.792      0.000       0.038       0.119
rain           0.4005      0.031     13.106      0.000       0.341       0.460
snow       -1.757e-15    1.1e-17   -159.294      0.000   -1.78e-15   -1.74e-15
thunder       -0.3411      0.029    -11.588      0.000      -0.399      -0.283
tornado        0.0103      0.021      0.499      0.618      -0.030       0.051
visibility     0.1844      0.022      8.516      0.000       0.142       0.227
wdirdegrees   -0.5271      0.022    -24.105      0.000      -0.570      -0.484
windspeed      0.4053      0.023     17.465      0.000       0.360       0.451
==============================================================================
Omnibus:                     1596.598   Durbin-Watson:                   0.528
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8896.300
Skew:                           1.162   Prob(JB):                         0.00
Kurtosis:                       8.505   Cond. No.                     2.92e+16
==============================================================================
```

Figure 17: Initial OLS Summary

Based on the p-values and **t-values of the coefficients**, I regressed backward to get the final model.
Further moving, the following columns were removed from the dataset based on their p-values. 'tornado', 'hail', 'pressure', 'visibility', 'snow', 'thunder', 'rain', 'fog'.

The final OLS result summary and the SVD analysis are as follows:

```
SingularValues = [1.22285722e+04 1.06454797e+04 8.04873134e+03 5.99215515e+03
 5.98000000e+03 5.97709383e+03 5.97417576e+03 5.29491741e+03
 4.41994423e+03 3.78477898e+03 1.97083742e+03 1.44331403e+03
 6.56243251e-13]
                            OLS Regression Results
==============================================================================
Dep. Variable:                   temp   R-squared:                       0.951
Model:                            OLS   Adj. R-squared:                  0.951
Method:                 Least Squares   F-statistic:                 2.914e+04
Date:                Wed, 10 May 2023   Prob (F-statistic):               0.00
Time:                        18:12:27   Log-Likelihood:                -11492.
No. Observations:                5980   AIC:                         2.299e+04
Df Residuals:                    5975   BIC:                         2.303e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
constant      24.8807      0.021   1163.299      0.000      24.839      24.923
dewpoint       6.7483      0.023    290.785      0.000       6.703       6.794
humidity      -5.5443      0.024   -233.332      0.000      -5.591      -5.498
wdirdegrees   -0.5620      0.023    -24.911      0.000      -0.606      -0.518
windspeed      0.4026      0.022     18.195      0.000       0.359       0.446
==============================================================================
Omnibus:                     1682.046   Durbin-Watson:                   0.471
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7311.155
Skew:                           1.313   Prob(JB):                         0.00
Kurtosis:                       7.738   Cond. No.                         1.66
==============================================================================
```

Figure 18: Final OLS Results Summary

From the final results we can see that on removing the above said columns, all the singular values are much higher than zero and condition number is 1.66. Both these factors indicate that there is not collinearity left in the dataset. Adjusted R^2 value is also not getting compromised with a value of 0.951 which means that 95.1% variance in the dataset can be explained by the OLS model.

F-TEST ANALYSIS

The F statistic p-value is less than 0.5, hence we can reject the NULL hypothesis, which means our model is better than intercept only model.

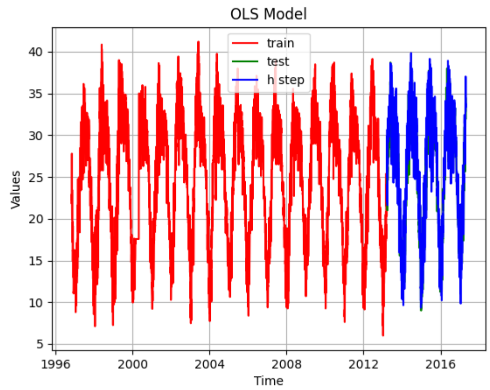


Figure 19: Predictions using OLS on Test Set

From the plot above we can se that the predicted values are totally coinciding with test set which means that the OLS model is performing very well on the test set.

```
MSE for OLS model: 1.82
RMSE for OLS model: 1.35
MAE for OLS model: 1.06
Q Value of OLS residuals: 42426.56
Mean of residuals for OLS: -4.1444365590823234e-15
Variance of residuals for OLS: 2.7332638210928875
```

Figure 20: Performance Metrics for OLS

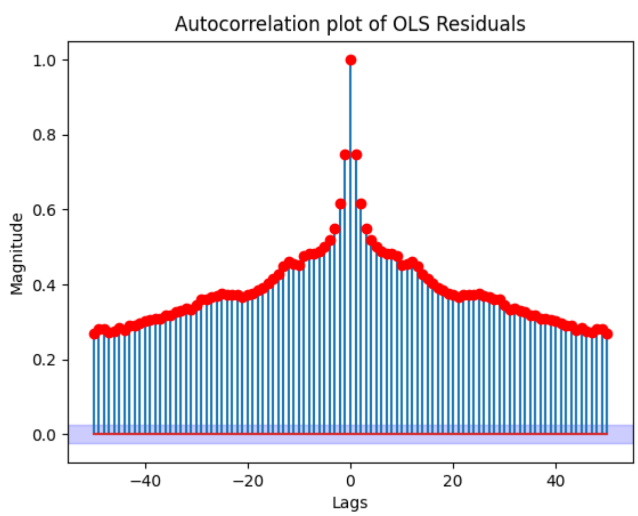


Figure 21: ACF Plot of OLS Residuals

From the performance metrics figure, we can see that OLS is performing very well for the dataset. However, the Q-value of OLS residuals are very high. ACF plot of OLS residuals also suggests that residuals are not white, which means that OLS has not captured the model very well. The reason why OLS is performing very well on the test set might be the overfitting.

BASE MODELS

To compare the ARIMA model, we need to build the base models as benchmarks.

Following models are used as benchmarks:

1. Average
2. Naïve
3. Drift
4. Simple and Exponential Smoothing

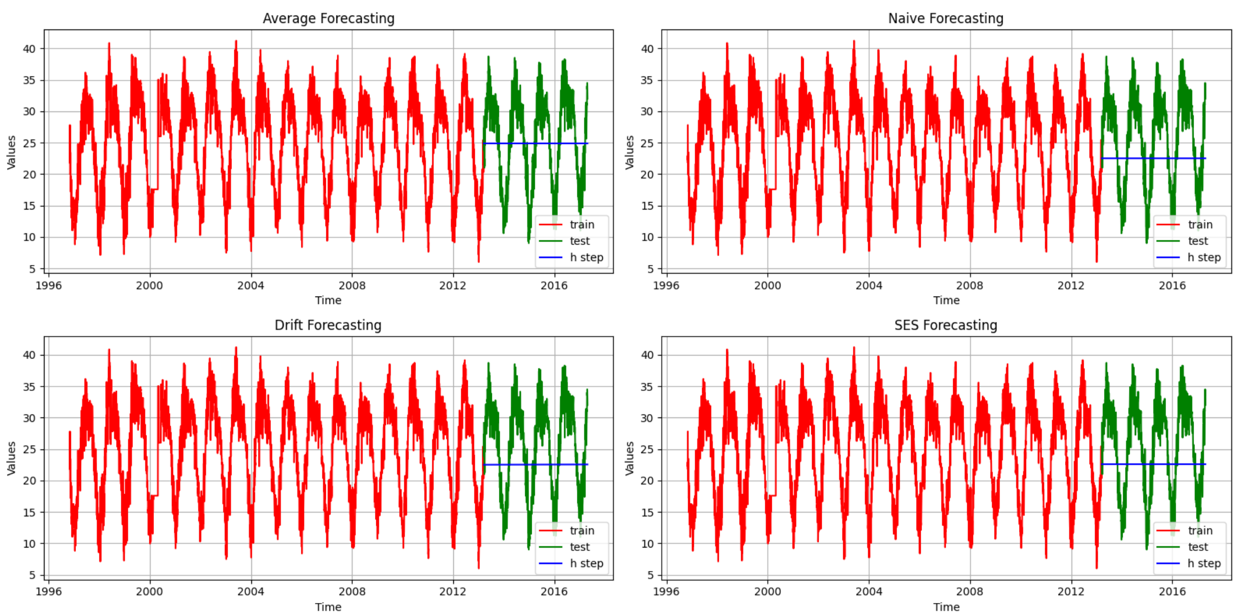


Figure 22: All the Base Models Performances on Test Set

```
MSE for Average forecasting: 51.64
RMSE for Average forecasting: 7.19
MAE for Average forecasting: 6.32
```

Figure 23: Performance Metrics for Average Forecasting

```
MSE for Naive forecasting: 61.19
RMSE for Naive forecasting: 7.82
MAE for Naive forecasting: 6.95
```

Figure 24. Performance Metrics for Naive Forecasting

```
MSE for Drift forecasting: 61.06
RMSE for Drift forecasting: 7.81
MAE for Drift forecasting: 6.95
```

Figure 25. Performance Metrics for Drift Forecasting

```
MSE for SES forecasting: 60.66
RMSE for SES forecasting: 7.79
MAE for SES forecasting: 6.93
```

Figure 26. Performance Metrics for SES Forecasting

From the figures above, we can see that all the base models are not able to perform very well on the test set and their performance metrics are almost like each other.

ARIMA MODEL ORDER DETERMINATION

When I fed the stationary time series to the GPAC function, following GPAC table was observed.

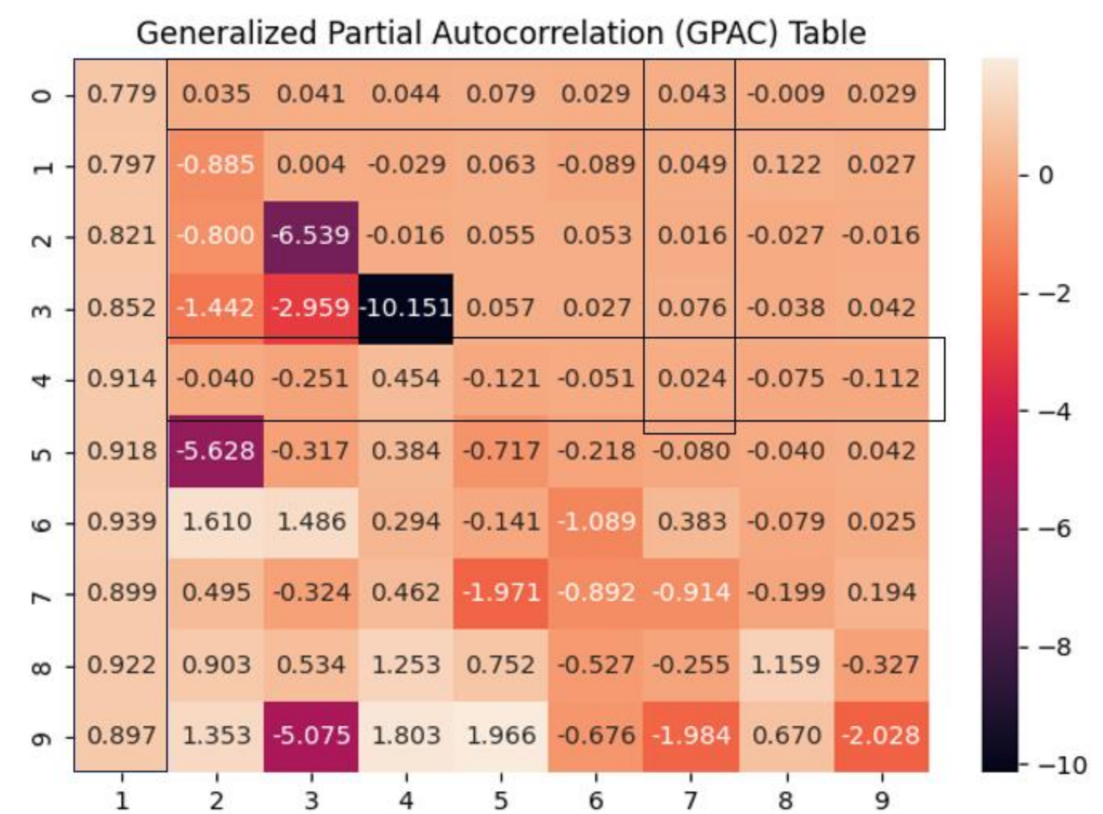


Table 4. GPAC Table

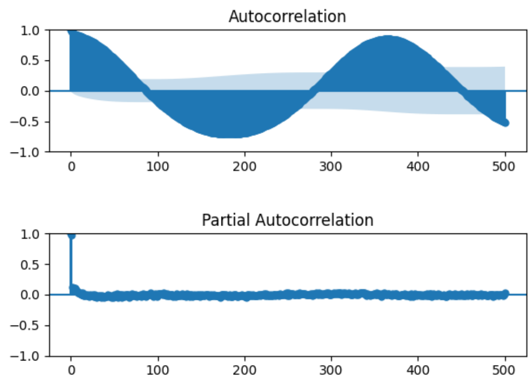


Figure 27. ACF with 500 lags

The ACF/PACF plot does not give the order which we got from the GPAC table but ACF/PACF plot suggests that we are dealing with AR process.

From the GPAC, we can see that there are three orders of ARMA possible.

1. (1, 0)

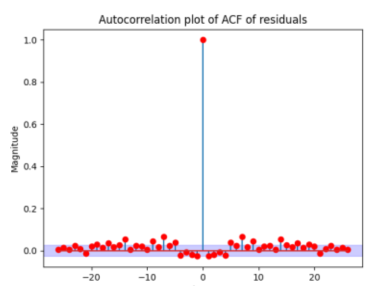


Figure 28. ACF when order is (1, 0)

```
Chi critical: 44.31410489621915
Q Value: 108.79023430024947
Alfa value for 99% accuracy: 0.01
The residual is NOT white
```

Figure 29. Check of Whiteness when order is (1, 0)

2. (1, 4)

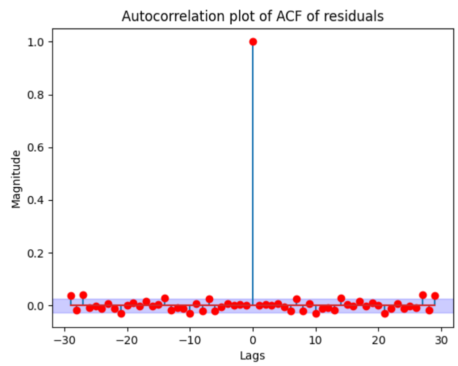


Figure 30. ACF when order is (1, 4)

```
Chi critical: 42.97982013935165
Q Value: 49.42935951699457
Alfa value for 99% accuracy: 0.01
The residual is NOT white
```

Figure 31. Check of Whiteness when order is (1, 4)

3. (7, 0) (although GPAC is not as clear as it should be, but it gave white noise of residuals)

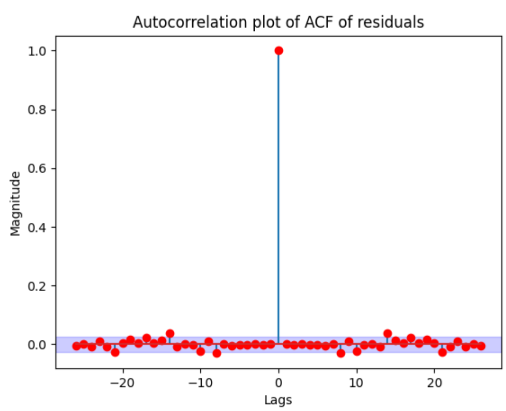


Figure 32. ACF when order is (7, 0)

```
Chi critical: 36.19086912927004
Q Value: 28.816840095729233
Alfa value for 99% accuracy: 0.01
The residual is white
```

Figure 33. Check of Whiteness when order is (7, 0)

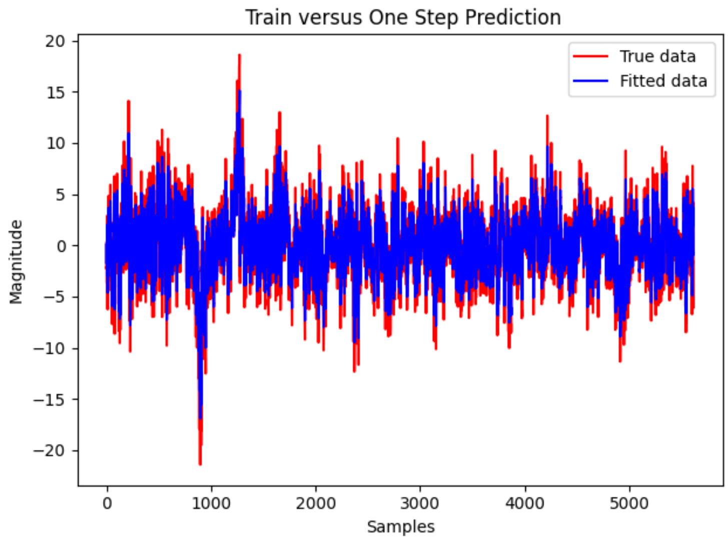


Figure 34. One Step prediction for ARIMA (7, 0, 0)

From the above, one step prediction plot, we can see that order (7, 0) is working very well and residuals are white. Hence, going with the order of ARMA (7, 0).

PARAMETER ESTIMATION USING LM ALGORITHM

Coefficients of ARMA model are found using Levenberg Marquardt algorithm. The following are the screenshots of the results.

```
Estimated parameters
[-7.42829683e-01 -5.50170484e-04 -7.91572802e-03  1.47823539e-02
 -5.68301399e-02  3.50946595e-03 -4.28426822e-02]

Estimated parameters with 95% confidence interval
-0.7695242246933348 < a1 < -0.7161351420821597
-0.03381337241023535 < a2 < 0.03271303144321823
-0.04114409319899834 < a3 < 0.025312637158452517
-0.018449114254206943 < a4 < 0.04801382205469202
-0.0900699472559529 < a5 < -0.023590332527007776
-0.02976630872532906 < a6 < 0.03678524063224158
-0.06955016564686381 < a7 < -0.016135198785105616
```

Figure 35. Parameter Estimation using LM with confidence intervals

From the parameter estimated, we can see that a_2, a_3, a_4, and a_6 are insignificant as the confidence interval of these parameters contains zero in it.

```
Estimated variance
[[5.1987423]]

Estimated standard deviation
[[2.28007506]]
```

Figure 36. Estimated Variance and Standard Deviation

DIAGNOSTIC ANALYSIS

Diagnostic Tests

- Confidence Intervals

```
Estimated parameters with 95% confidence interval
-0.7695242246933348 < a1 < -0.7161351420821597
-0.033813372410235354 < a2 < 0.03271303144321823
-0.04114409319899834 < a3 < 0.025312637158452517
-0.018449114254206943 < a4 < 0.04801382205469202
-0.0900699472559529 < a5 < -0.023590332527007776
-0.02976630872532906 < a6 < 0.03678524063224158
-0.06955016564686381 < a7 < -0.016135198785105616
```

Figure 37. Estimated parameters Confidence Interval

From the parameter estimated, we can see that a_2, a_3, a_4, and a_6 are insignificant as the confidence interval of these parameters contains zero in it.

- Zero-Pole cancellation

```
Roots of the denominator are: [ 0.89688509  0.47812317  0.47812317 -0.50765058 -0.50765058 -0.04750029
 -0.04750029]
```

Since, this is an AR process, so numerator will not have roots hence there will be no zero-pole cancellation.

- Chi-Square Test

```
Chi critical: 36.19086912927004
Q Value: 28.816840095729233
Alfa value for 99% accuracy: 0.01
The residual is white
```

Figure 38. Chi-Square Test of the Residuals

- Estimated Covariance and Variance of Estimated Parameters

```
Estimated covariance matrix
[[ 1.78149634e-04 -1.32535627e-04 -5.14718999e-07 -1.19427468e-06
   2.44792676e-06 -1.01996877e-05 -5.02371624e-06]
 [-1.32535627e-04  2.76610151e-04 -1.32438193e-04  4.52672370e-07
  -3.06108944e-06  1.00152037e-05 -1.01962135e-05]
 [-5.14718999e-07 -1.32438193e-04  2.76031063e-04 -1.32280979e-04
   3.60638083e-07 -3.07103151e-06  2.44963493e-06]
 [-1.19427468e-06  4.52672370e-07 -1.32280979e-04  2.76082619e-04
  -1.32374345e-04  4.01820537e-07 -1.19614620e-06]
 [ 2.44792676e-06 -3.06108944e-06  3.60638083e-07 -1.32374345e-04
   2.76221198e-04 -1.32475852e-04 -5.80167485e-07]
 [-1.01996877e-05  1.00152037e-05 -3.07103151e-06  4.01820537e-07
  -1.32475852e-04  2.76819295e-04 -1.32628510e-04]
 [-5.02371624e-06 -1.01962135e-05  2.44963493e-06 -1.19614620e-06
  -5.80167485e-07 -1.32628510e-04  1.78322418e-04]]

Estimated variance
[[5.1987423]]
```

Figure 39. Estimated Covariance and Variance

- Since the ACF of residuals suggest that residuals are white, it means that the estimator is unbiased as it has extracted all the information and is generalizing well for the dataset.
- Variance of Residual and Forecast Error

```
Variance of residual error: 5.2
Variance of forecast error: 11.68
```

Figure 40. Variance of Residual and Forecast Error

- Performance of ARIMA on TEST set

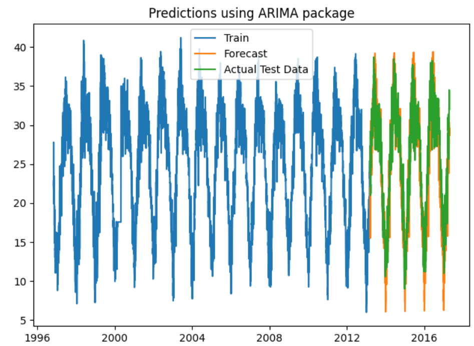
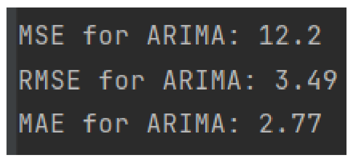


Figure 41. Performance of ARIMA on Test Set



```
MSE for ARIMA: 12.2
RMSE for ARIMA: 3.49
MAE for ARIMA: 2.77
```

Figure 42. Performance Metric for ARIMA

DEEP LEARNING MODEL

To apply deep learning models for this dataset, LSTM is used.

- Two-layer LSTM networks were used.
- The first layer contains the activation function as 'Relu' with a total of 64 neurons.
- The second layer contains 50 neurons.
- A total of 10 epochs were run on the personal computer. (Due to computational limitations, epochs could not be increased more than 10).

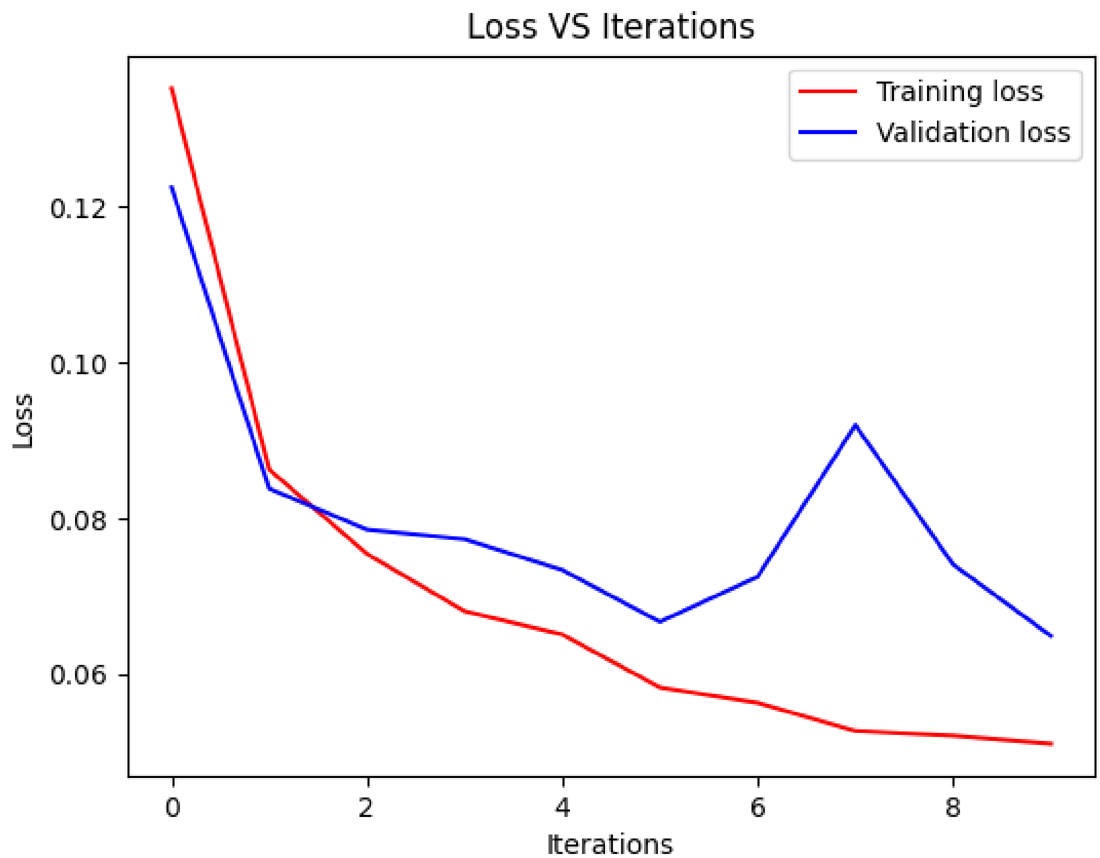


Figure 43. Loss VS Iteration Plot

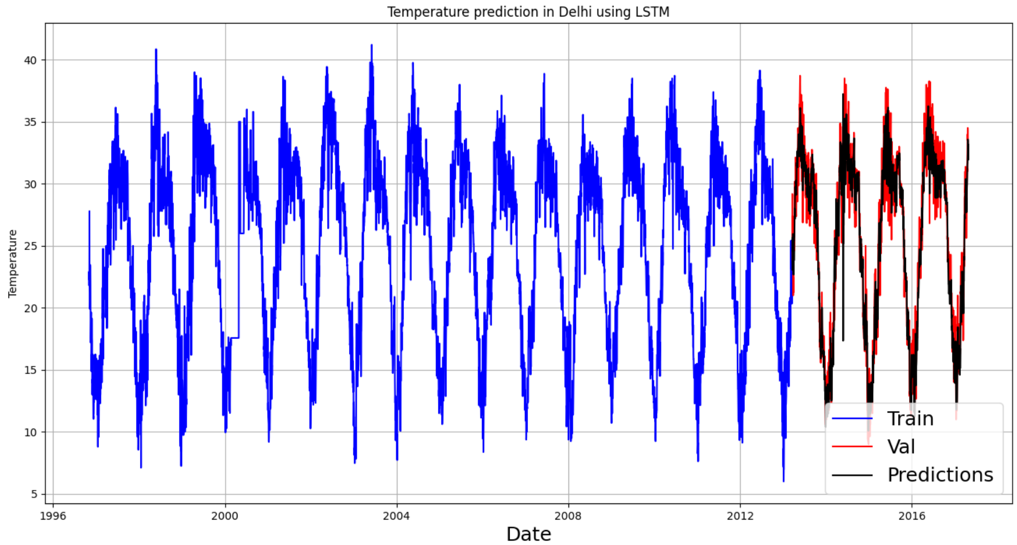


Figure 44. Predictions using LSTM on Test Set

```
MSE for LSTM model: 3.08
RMSE for LSTM model: 1.75
MAE for LSTM model: 1.34
```

Figure 45. Performance Metric for LSTM

FINAL MODEL SELECTION

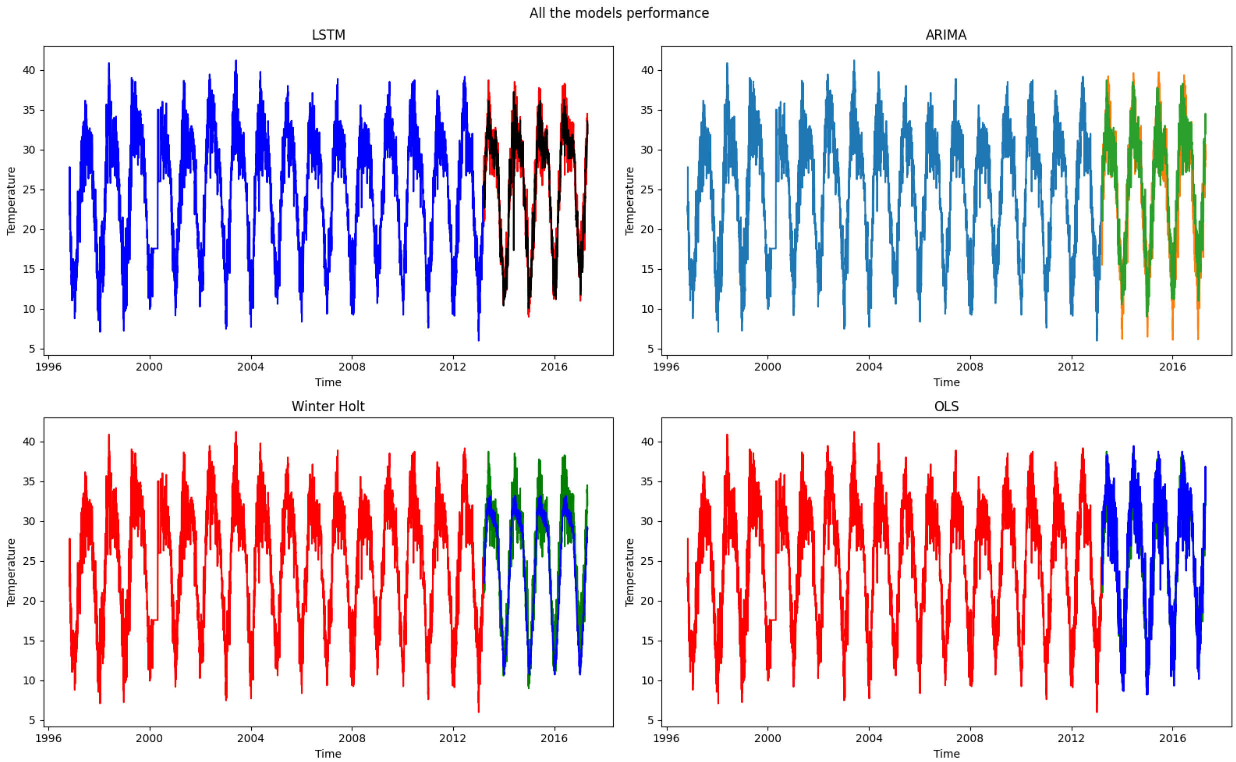


Figure 46. Performance Comparison of all Model on Test Set

```
MSE for LSTM model: 3.08
RMSE for LSTM model: 1.75
MAE for LSTM model: 1.34
```

```
MSE for ARIMA: 12.2
RMSE for ARIMA: 3.49
MAE for ARIMA: 2.77
```

```
MSE for Average forecasting: 51.64
RMSE for Average forecasting: 7.19
MAE for Average forecasting: 6.32
```

```
MSE for Naive forecasting: 61.19
RMSE for Naive forecasting: 7.82
MAE for Naive forecasting: 6.95
```

```
MSE for Drift forecasting: 61.06
RMSE for Drift forecasting: 7.81
MAE for Drift forecasting: 6.95
```

```
MSE for SES forecasting: 60.66
RMSE for SES forecasting: 7.79
MAE for SES forecasting: 6.93
```

```
MSE for Winter-Holt method: 7.63
RMSE for Winter-Holt method: 2.76
MAE for Holt-Winter method: 2.19
```

```
MSE for OLS model: 1.82
RMSE for OLS model: 1.35
MAE for OLS model: 1.06
Q Value of OLS residuals: 42426.56
Mean of residuals for OLS: -4.1444365590823234e-15
Variance of residuals for OLS: 2.7332638210928875
```

Figure 47. Performance Metric Comparison

Based on all the metrics for all the models above in the report, OLS model has the least MSE, RMSE, and MAE as compared to all other models. But the Q value of the residuals of OLS model were very high, which indicates that OLS model did not extract all the information from the time series. The very low MSE is probably due to overfitting of the OLS model.

So, keeping this in mind, LSTM model worked best out of all the models tried and it generalized very well with the time series.

FORECAST FUNCTION

The equation of the ARIMA model developed is:

$$y_t - 0.7428y_{t-1} - 0.0568y_{t-5} - 0.0428y_{t-7} = e_t$$

Equation 3. ARIMA Model Equation

Forecast Equations:

1 Step: $\boldsymbol{\hat{y}_t(1) = 0.7428y_t + 0.0568y_{t-4} + 0.0428y_{t-6}}$

Equation 4. 1 step Prediction

2 Step: $\boldsymbol{\hat{y}_t(2) = 0.7428\hat{y}_t(1) + 0.0568y_{t-3} + 0.0428y_{t-5}}$

Equation 5. 2 step Prediction

3 Step: $\boldsymbol{\hat{y}_t(3) = 0.7428\hat{y}_t(2) + 0.0568y_{t-2} + 0.0428y_{t-4}}$

Equation 6. 3 step Prediction

4 Step: $\boldsymbol{\hat{y}_t(4) = 0.7428\hat{y}_t(3) + 0.0568y_{t-1} + 0.0428y_{t-3}}$

Equation 7. 4 step Prediction

5 Step: $\boldsymbol{\hat{y}_t(5) = 0.7428\hat{y}_t(4) + 0.0568y_t + 0.0428y_{t-2}}$

Equation 8. 5 step Prediction

6 Step: $\hat{y}_t(6) = 0.7428\hat{y}_t(5) + 0.0568\hat{y}_t(1) + 0.0428y_{t-1}$

Equation 9. 6 step Prediction

7 Step: $\hat{y}_t(7) = 0.7428\hat{y}_t(6) + 0.0568\hat{y}_t(2) + 0.0428y_t$

Equation 10. 7 step Prediction

h Step: $\hat{y}_t(h) = 0.7428\hat{y}_t(h-1) + 0.0568\hat{y}_t(h-5) + 0.0428\hat{y}_t(h-7)$

Equation 11. h step Prediction

H-STEP AHEAD PREDICTION

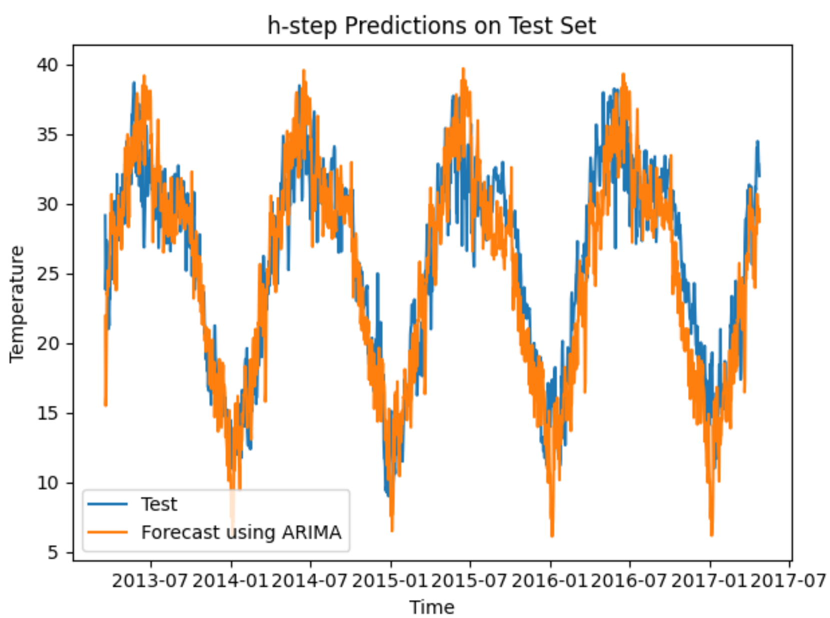
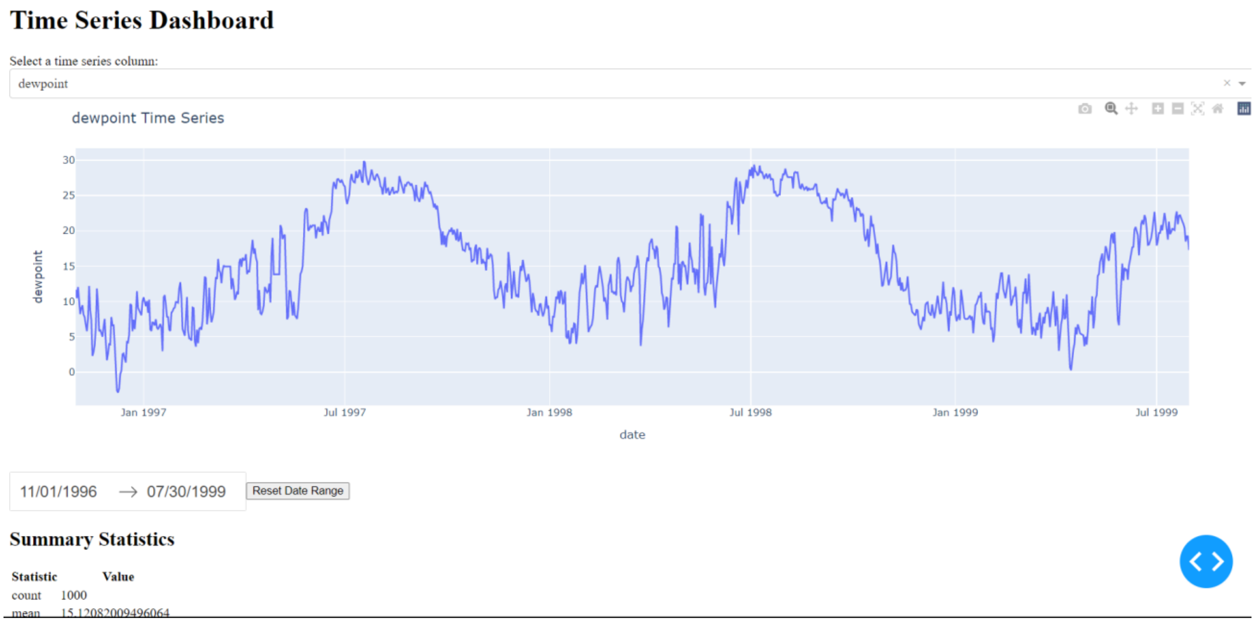


Figure 48. h-step Predictions using Custom ARIMA Forecast Function

Custom forecast function of ARIMA is also generalizing very well on the test as we can see from the plot above.

PYTHON DASH DASHBOARD



This is an interactive dashboard where we can select the column from the drop down and time series for that column will be displayed. We can select the dates from the UI. Summary statistics are also being displayed for that particular column.

SUMMARY AND CONCLUSION

- The Delhi weather time series is highly seasonal.
- LSTM model worked very well for the dataset.
- Base models did not work well for the dataset as they did not incorporate seasonality in them.
- The Winter-Holts method is a very good method to forecast seasonal data.
- Interpolation and Errors: The utilization of interpolation to fill in missing values can lead to errors that could impact the accuracy of predictions.
- The data was highly seasonal, so SARIMA would have been a better choice, but due to computational needs of the process, it did not work here.

REFERENCES

- statsmodels.regression.linear_model.OLS - statsmodels 0.15.0 (+6). (n.d.). https://www.statsmodels.org/devel/generated/statsmodels.regression.linear_model.OLS.html
- statsmodels.tsa.holtwinters.ExponentialSmoothing - statsmodels 0.15.0 (+6). (n.d.). https://www.statsmodels.org/devel/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html

- Forecasting: Principles and Practice (3rd ed). (n.d.). https://otexts.com/fpp3/

APPENDIX

- Forecast function:

```python
def forecast(y, T, h):
    T = T - 1
    y_hat = []
    for i in range(1, h+1):
        if i == 1:
            y_hat.append(0.7428 * y[T] + 0.0568 * y[T - 4] + 0.0428 * y[T - 6])
        elif i == 2:
            y_hat.append(0.7428 * y_hat[0] + 0.0568 * y[T - 3] + 0.0428 * y[T - 5])
        elif i == 3:
            y_hat.append(0.7428 * y_hat[1] + 0.0568 * y[T - 2] + 0.0428 * y[T - 4])
        elif i == 4:
            y_hat.append(0.7428 * y_hat[2] + 0.0568 * y[T - 1] + 0.0428 * y[T - 3])
        elif i == 5:
            y_hat.append(0.7428 * y_hat[3] + 0.0568 * y[T] + 0.0428 * y[T - 2])
        elif i == 6:
            y_hat.append(0.7428 * y_hat[4] + 0.0568 * y_hat[0] + 0.0428 * y[T - 1])
        elif i == 7:
            y_hat.append(0.7428 * y_hat[5] + 0.0568 * y_hat[1] + 0.0428 * y[T])
        else:
            y_hat.append(0.7428 * y_hat[i-1-1] + 0.0568 * y_hat[i - 5 - 1] + 0.0428 * y[i - 7 - 1])
    y_hat = np.array(y_hat)
    return y_hat
```

Figure 49. Forecast Function for ARIMA Model

P.S. All other codes like differencing, LM, and GPAC were developed for the labs in the course.

- Reverse Transform function from Seasonally differenced series

```
def reverse_transform_and_plot(prediction, y_train, y_test, title):
    forecast = []
    s = 365
    for i in range(len(y_test)):
        if i < s:
            forecast.append(prediction[i] + y_train[- s + i])
        else:
            temp = i - s
            forecast.append(prediction[i] + forecast[temp])
    forecasted_values = pd.Series(forecast)
    forecasted_values.index = prediction.index
    plt.plot(y_train.index, y_train.values, label='Train')
    plt.plot(forecasted_values.index, forecasted_values.values, label='Forecast')
    plt.plot(y_test.index, y_test.values, label='Actual Test Data')
    str = f'Predictions using {title}'
    plt.title(str)
    plt.legend()
    plt.tight_layout()
    plt.show()

    return forecast
```