BRAZIL HUMIDITY PREDICTION

Anjali Mudgal

ABSTRACT

Predicting Relative Humidity at station ID 'A010' in North Brazil by analyzing temperature, precipitation, wind speed, and atmospheric pressure for accurate climate monitoring and forecasting.

Mudgal, Anjali

Time Series Forecasting

Table of Contents

Table of Figure

Table of Equation

Table of Code

Abstract

This project focuses on predicting the Relative Humidity in North Brazil, specifically at station ID 'A010'. Relative humidity is influenced by various factors, including temperature, precipitation, wind speed, and atmospheric pressure. Understanding the interplay of these factors is crucial for accurate humidity predictions and gaining insights into seasonal patterns.

Temperature has a significant impact on relative humidity, as warm air can hold more moisture than cool air. Additionally, precipitation, such as rainfall, introduces moisture into the air, affecting humidity levels. Wind speed plays a role in humidity by causing moisture to evaporate more quickly under strong winds. On the other hand, calm conditions allow moisture to linger, resulting in higher humidity levels. Furthermore, atmospheric pressure is associated with humidity, with low-pressure systems often indicating high humidity and high-pressure systems indicating lower humidity percentages.

Through comprehensive time series analysis, considering temperature, precipitation, wind speed, and atmospheric pressure, this study aims to uncover seasonal variations and make accurate predictions of Relative Humidity in North Brazil at station ID 'A010'. The findings from this research will contribute to a better understanding of the factors influencing humidity patterns in the region, enabling more effective climate monitoring and forecasting.

Introduction

In this report, we delve into the analysis of a Brazil humidity dataset and explore various factors that can potentially influence it. Our objective is to employ different models, such as average, drift, naïve, Simple Exponential, Holt Winter, ARMA, Linear Regression, and LSTM, to predict humidity levels. The primary aim is to compare these models and identify the most suitable one that effectively represents the data and facilitates accurate predictions.

To determine the ARMA model, we employ several techniques, including assessing stationarity, analyzing the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF), and employing the Generalized Partial Autocorrelation (GPAC) method. Through these steps, we aim to determine the optimal order and equation that best capture the characteristics of the data.

To obtain statistically significant coefficients that accurately represent the data, we utilize the Levenberg Marquardt Algorithm. This algorithm helps us estimate the coefficients that align

with the observed data and are deemed statistically significant. These coefficients serve as vital inputs for our prediction model.

Subsequently, using the selected model and obtained information, we conduct predictions for relative humidity. By evaluating the mean square error and performing diagnostic analyses on the residuals, we rigorously assess the model's performance. These evaluations guide us in selecting the most appropriate model from the candidate models for our specific dataset.

Through this comprehensive analysis and model comparison, we aim to identify the optimal model for predicting relative humidity in the given Brazil dataset. The chosen model will not only provide accurate predictions but also enable a deeper understanding of the underlying patterns and factors influencing humidity levels.

Description of the dataset

I have Climate Weather Surface of Brazil - Hourly Kaggle dataset for my project. Here we have the following features :

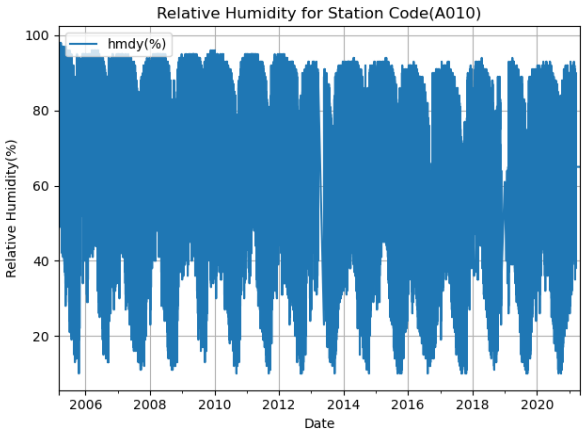
- Date (YYYY-MM-DD)
- Time (HH:00)
- Amount of precipitation in millimetres (last hour)
- Atmospheric pressure at station level (mb)
- Maximum air pressure for the last hour (mb)
- Minimum air pressure for the last hour (mb)
- Solar radiation (KJ/m2)
- Air temperature (instant) (°c)
- Dew point temperature (instant) (°c)
- Maximum temperature for the last hour (°c)
- Minimum temperature for the last hour (°c)
- Maximum dew point temperature for the last hour (°c)
- Minimum dew point temperature for the last hour (°c)
- Maximum relative humid temperature for the last hour (%)
- Minimum relative humid temperature for the last hour (%)
- Relative humid (% instant)
- Wind direction (radius degrees (0-360))
- Wind gust in metres per second
- Wind speed in metres per second
- Brazilian geopolitical regions
- State (Province)
- Station Name (usually city location or nickname)
- Station code (INMET number)
- Latitude
- Longitude
- Elevation

Figure 1 Dataset features

For this time series analysis, we focus on station code 'A010' in North Brazil. Latitude, longitude, and elevation variations influence humidity levels. By examining this dataset, we uncover temporal patterns and understand regional humidity dynamics. This analysis aids local climate monitoring and prediction efforts. Here we can ‘A010’ lies in north-eastern region of Brazil.

Figure 2 Station location of the selected dataset

With this I would be dropping my location dependent features such as 'region', 'station', 'state', 'latitude', 'longitude', 'height', 'station code'.



This is what the initial time series looks like with hourly data (141648 observations).

Pre-processing dataset :

Daily aggregation

```
>>> df.isna().sum()
prcp(mm)          19577
atmp(mb)          10708
atmmax            10861
atmmin            10861
radi(KJ/m2)       71785
temp(C)           12432
dewp(C)           12432
tmax              12559
tmin              12558
dmax              12562
dmin              12562
hmax(%)           10785
hmin              10820
hmdy(%)           10731
wdir(deg)         11370
wgust(m/s)        11497
wdsp(m/s)         11369
```

```
>>> daily_df.isna().sum()
prcp(mm)      622
atmp(mb)      252
atmmax        263
atmmin        263
radi(KJ/m2)   265
temp(C)       253
dewp(C)       253
tmax          261
tmin          261
dmax          261
dmin          261
hmax(%)       260
hmin          260
hmdy(%)       252
wdir(deg)     279
wgust(m/s)    285
wdsp(m/s)     279
```

Figure 3 Missing Values before and After aggregating data on daily basis

Since my dataset has a lot of missing value, I am doing a daily aggregation with mean of hourly data.

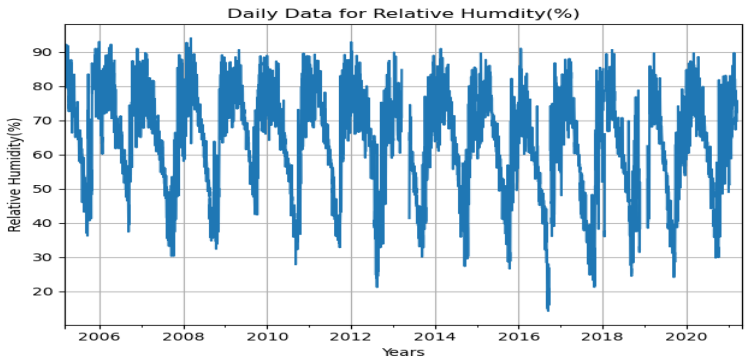


Figure 4 Target Variable(Relative Humidity) Series after daily aggregation

Filling the missing values

Since the missing values have now been reduced, I am using spline interpolation to fill in those values. The spline curve consists of multiple polynomial segments, typically cubic polynomials, that smoothly connect the data points.

Cleaned Data

Plot of Dependent Variable(Relative Humidity(%)) against time

This is how my Relative Humidity (%) looks like against time(daily) now. I have 5902 observations.

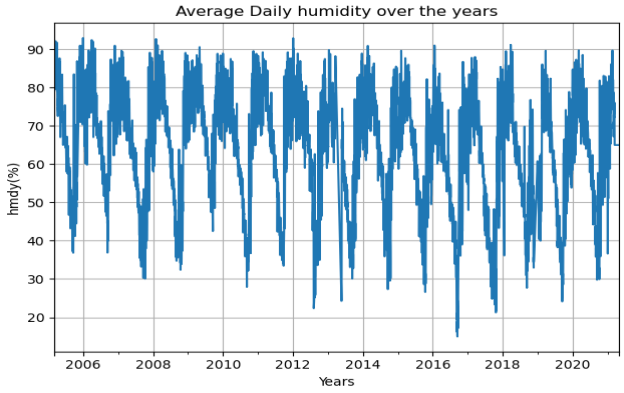


Figure 5 Relative Humidity after aggregation and filling missing values

ACF/PACF of Relative Humidity(%)

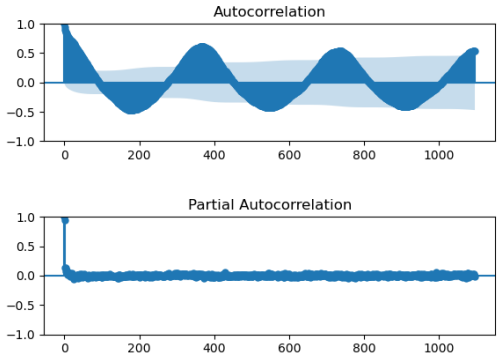


Figure 6 ACF/PACF of daily data(raw) for 1000 lags, here we can see the seasonality of 365

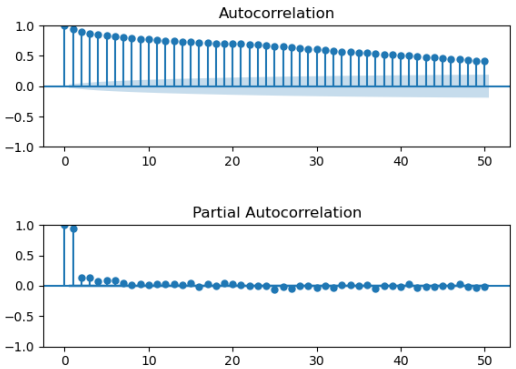


Figure 7 ACF/PACF of daily data with 50 lags

Here we can notice, the seasonality of 365 and an AR of order (1,0)

Correlation Matrix with Pearson Coefficients

This is how relative humidity(%) is correlated with other variables. As we can notice here, the high values of correlation coefficients are with hmin, hmax(these are max and min humdity), temp(C) which is temperature in Celsius, tmin(C), tmax(C). Dmin, dmax, dewp(dew point temperature min, max and instant in Celsius).

Also from other correlation in matrix we can notice some correlations.

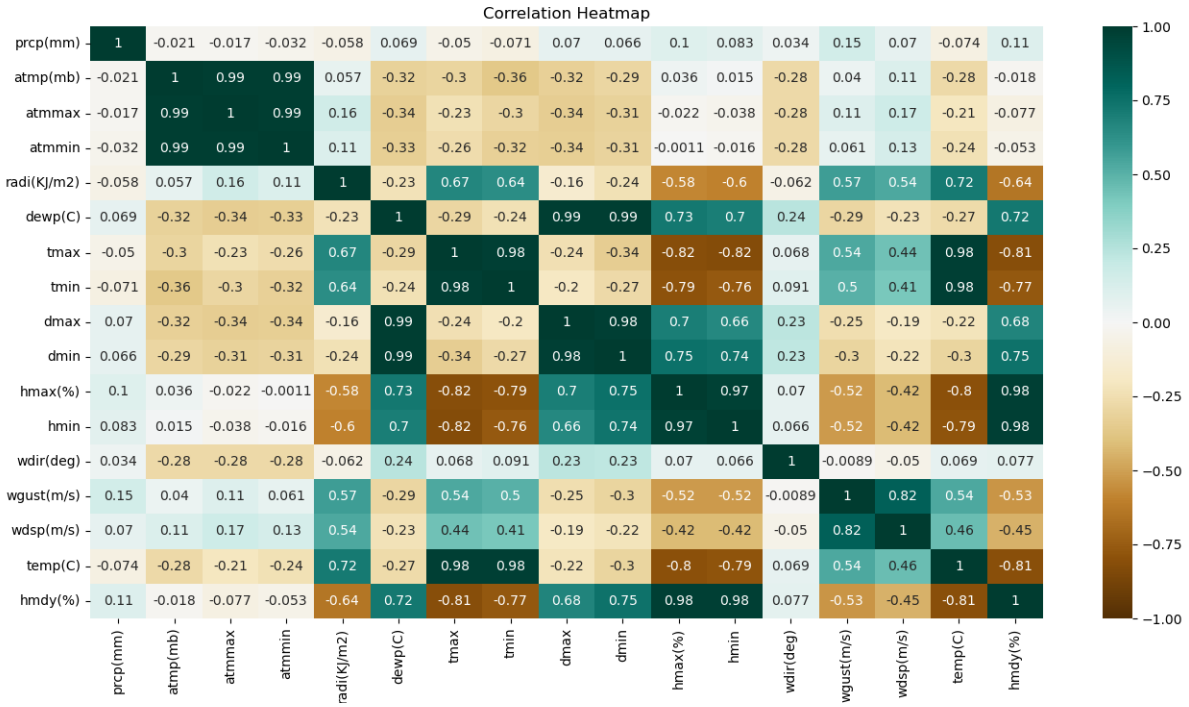


Figure 8 Correlation Matrix

Train Test Split

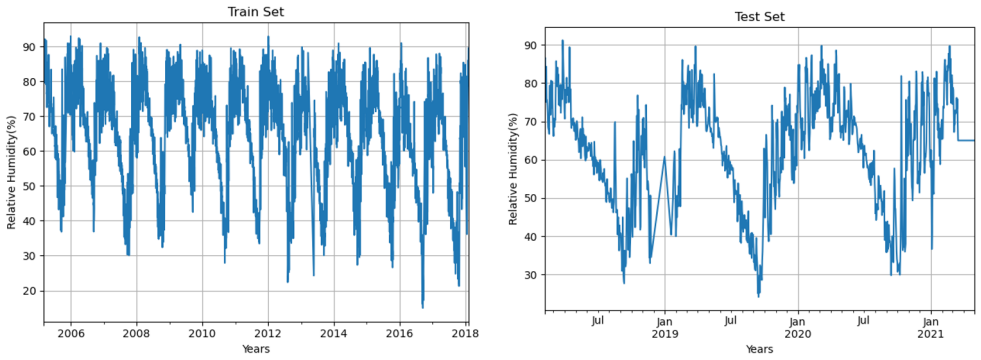


Figure 9 Train Test Split

The training dataset consists of 4721 observations, while the test set contains 1181 observations. It is worth noting that in the test set, some missing values have been filled using interpolation techniques. This interpolation method may lead to a slightly higher error than anticipated when evaluating the model's performance on the test data. It is important to consider this potential impact of interpolated values when interpreting the accuracy of the model's predictions on the test set.

Stationarity

To check the stationarity of raw data I would be using rolling mean, rolling variance, ADF and KPSS tests. As we can see from the results of rolling variance and KPSS test, the dataset is a marginally stationary.

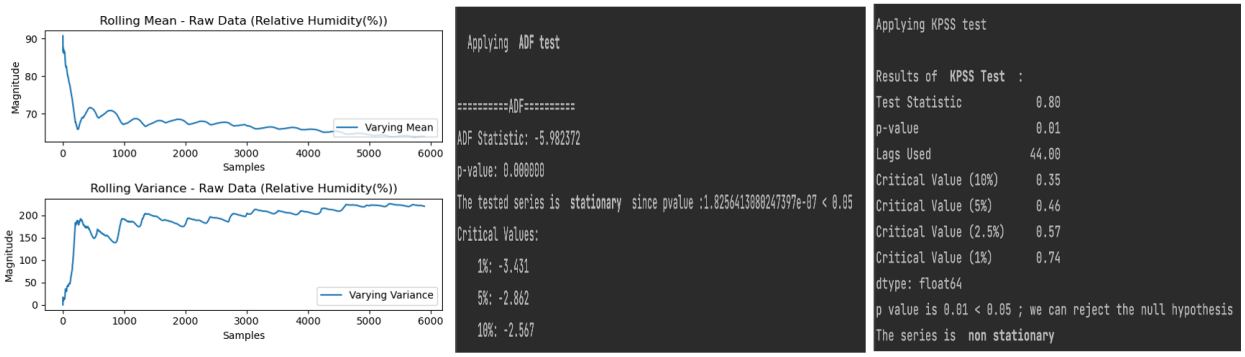


Figure 10 Stationarity Check on Original Data

By looking at the results and ACF/PACF , I am performing a seasonal differencing of 365.

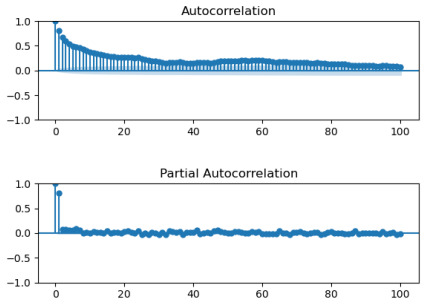


Figure 11 Original Data ACF/PACF

After seasonal differencing of 365.

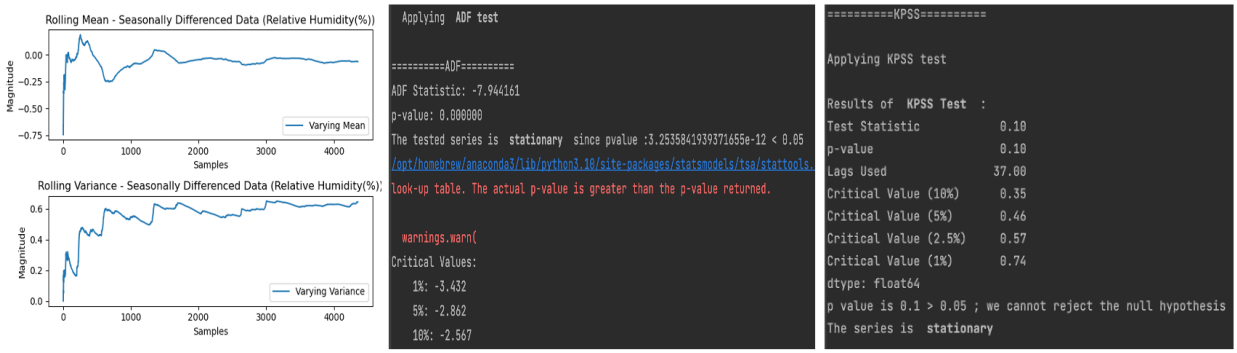


Figure 12 Stationarity Check on Seasonally Differenced data.

As we can see clearly here the series is stationary.

Time Series Decomposition

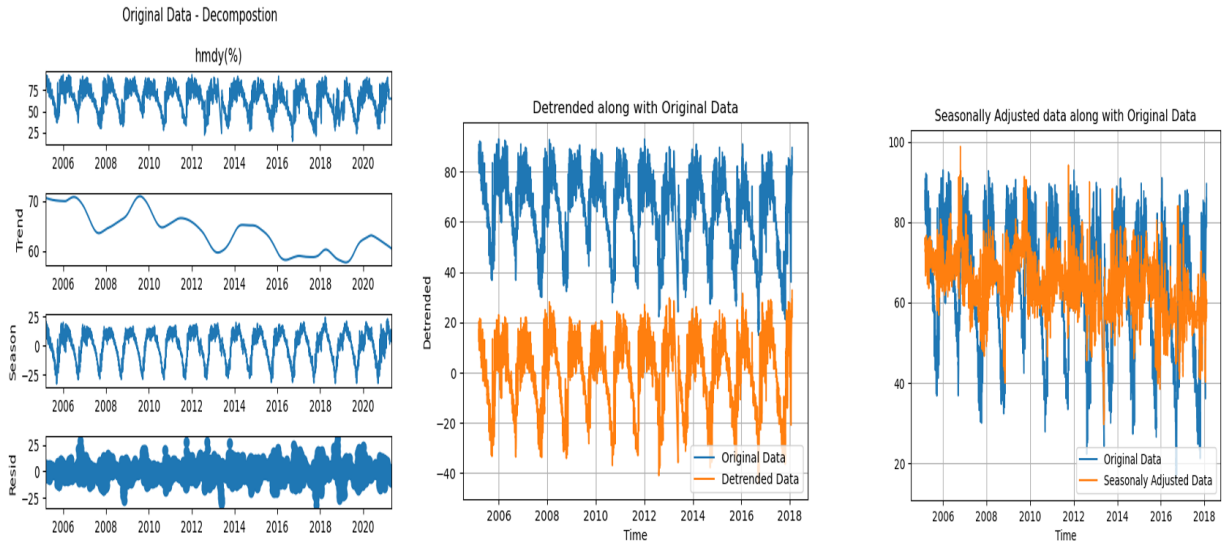


Figure 13 STL Decomposition

Equation 1 Strength of Seasonality and Strength of Trend

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

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

```
=====Strength of seasonality and Trend=====
The strength of trend for this data set is 0.33
```

```
The strength of seasonality for this data set is 0.80
```

Figure 14 Strength of Seasonality and Strength of Trend for our Dataset

As we can see the detrended and original data are almost similar and the value for trend is very low.
Also the high value of seasonality makes sense, since this dataset is highly seasonal.

From the STL decomposition we can see that the dataset is seasonal with the strength of seasonality as 0.81.

Base-models

Average

$$\hat{y}_{T+h|T} = \frac{y_1 + y_2 + \ldots + y_T}{T}$$

Equation 2 Average Prediction

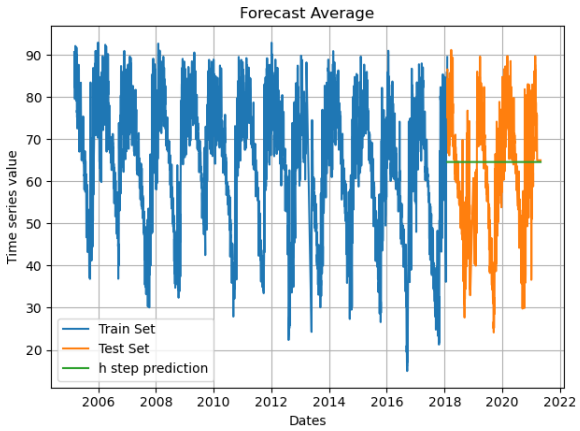


Figure 15 Forecast using Average Base Model

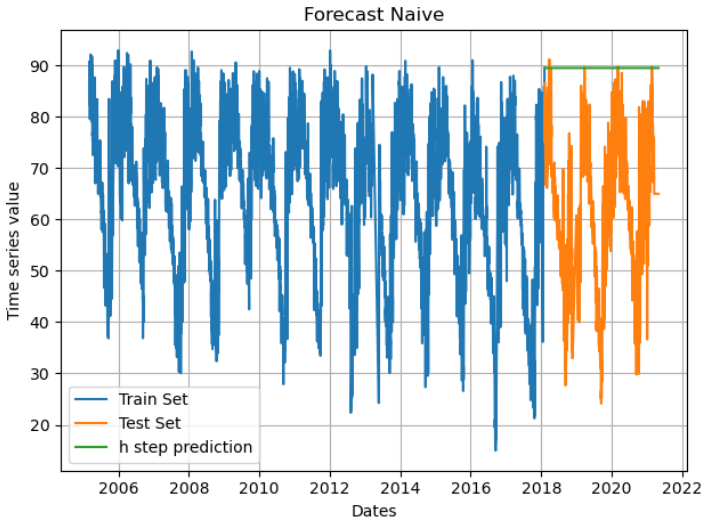
```
For Average
MSE for residual error is 223.34
RMSE for residual error is 14.94
```

Figure 16 MSE Average

Naïve

$$\hat{y}_{T+h|T} = y_T$$

Equation 3 Naive Forecast



Equation 4 Naive Forecast

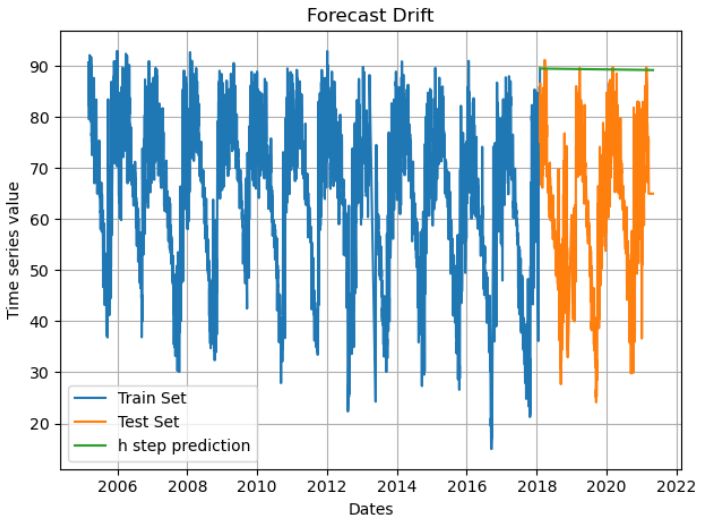
```
For Naive
MSE for residual error is 27.64
RMSE for residual error is 5.26
```

Figure 17 MSE Naive

Drift

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h(\frac{y_T - y_1}{T-1})$$

Equation 5 Drift Method



Equation 6 Forecast using Drift

```
For Drift
MSE for residual error is 27.68
RMSE for residual error is 5.26
```

Figure 18 MSE for Drift

Simple Exponential

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}$$

Equation 7 Simple Exponential

Best results are with alpha = 0.1.

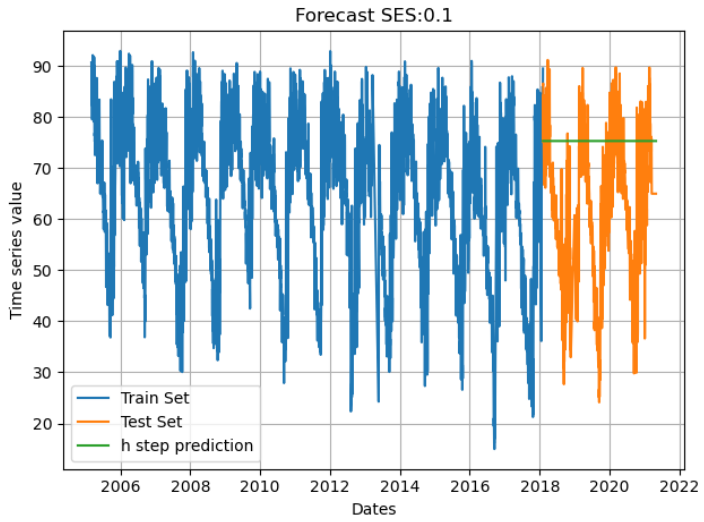


Figure 19 Simple Exponential Smoothing Prediction (alpha = 0.1)

```
MSE for residual error is 47.99
RMSE for residual error is 6.93
```

Figure 20 MSE for SES

Base models are performing bad for the test data and thus we will not use them.

Holt-Winters method

$$\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \\
k &\text{ is the integer part of } (h-1)/m,
\end{aligned}$$

Equation 8 Holt Winter Equation

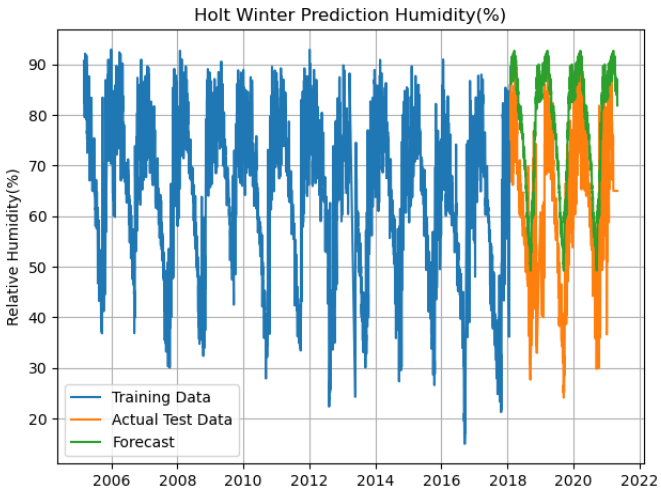


Figure 21 Holt Winter Prediction

Holt Winter Method is performing the best since predictions are close to the original test data, and here we can see the Root mean square error is 19

```
mean square error for Holt Winter is : 361.21
Root Mean Square Error Holt Winter is : 19.00
```

```
 Mean Forecast Error : -16.33
Variance Forecast Error:94.39
```

As we can see here that holt Winter is able to capture the dataset well and is giving good results for my datset. The reason it captures the seasonality well.

Multiple Linear Regression

Feature Elimination

Dropping Column

Since we are predicting Relative Humidity(%) (instant) for our dataset dropping minimum, maximum humidity as well.

SVD – Initial

```
=====SVD=====

Sigular Values = [2.46467449e+04 1.92860444e+04 7.42017802e+03 5.05142162e+03
 3.47054658e+03 2.93872261e+03 2.56107223e+03 6.24445945e+02
 4.90936052e+01 1.33944119e+01 8.06551462e+00 4.63195612e+00
 3.22225770e+00 2.41586075e+00]
```

In the SVD, we can see a lot of values are close to 0, which means there is some collinearity in the dataset.

VIF – Initial

Some of the VIF values are more than even 100, so I have eliminated the features one by one, after checkingt the vif values of the remaining features

```
=== VIF =====

      feature      VIF
     prcp(mm)     1.27
    wdir(deg)     1.67
  radi(KJ/m2)     2.66
    wdsp(m/s)     4.06
   wgust(m/s)     4.84
         tmax    99.41
         tmin   162.46
     temp(C)    237.74
         dmin   595.03
         dmax   615.19
       atmmax   717.66
       atmmin   790.91
     atmp(mb)   966.30
      dewp(C)  1286.47
```

Table 1 VIF Values Initial

VIF – Final

feature	VIF
prcp(mm)	1.22
radi(KJ/m2)	1.49
wdir(deg)	1.54
dmax	1.95
tmin	1.97
atmmax	2.15
wdsp(m/s)	3.73
wgust(m/s)	4.30

Table 2 VIF Values Final

SVD – Final

```
=====Singular Values=====

Sigular Values = [12003.9285836   9126.93197094  6531.04359867  3190.98120986
  2629.8265942   2395.35253886  1274.14166683   607.79383704]


=====Condition Number=====

Condition Number  4.444097264565293
```

Figure 22 SVD Final and Condition Number

Since the SVD values are now not close to 0 and also the condition number is very low we can see that there is no more collinearity left in the dataset.

```python
def featureRemovalVIF(train):
    train = train.loc[:, train.columns != 'hmdy(%)']
    vif_data = pd.DataFrame()

    vif_data["feature"] = train.columns
    vif_data["VIF"] = [variance_inflation_factor(train.values, i)
                       for i in range(len(train.columns))]

    vif = vif_data.sort_values(by=['VIF'])
    print(vif)
    while(vif['VIF'].iloc[-1] > 5):
        print(f"\nremoving {vif['VIF'].iloc[-1]}")
        train = train.loc[:, train.columns != vif['feature'].iloc[-1]]
        vif_data = pd.DataFrame()
        vif_data["feature"] = train.columns
        vif_data["VIF"] = [variance_inflation_factor(train.values, i)
                           for i in range(len(train.columns))]
        vif = vif_data.sort_values(by=['VIF'])
        print(vif)
    return vif['feature']
```

Code 1 VIF Feature Removal

Regression Analysis

OLS Regression Results

Dep. Variable:	hmdy(%)	R-squared:	0.951
Model:	OLS	Adj. R-squared:	0.951
Method:	Least Squares	F-statistic:	1.136e+04
Date:	Tue, 09 May 2023	Prob (F-statistic):	0.00
Time:	17:17:15	Log-Likelihood:	407.68
No. Observations:	4721	AIC:	-797.4
Df Residuals:	4712	BIC:	-739.2
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.242e-15	0.003	3.84e-13	1.000	-0.006	0.006
prcp(mm)	0.0635	0.004	17.788	0.000	0.057	0.071
radi(KJ/m2)	-0.0739	0.004	-18.732	0.000	-0.082	-0.066
wdir(deg)	0.0079	0.004	1.977	0.048	6.7e-05	0.016
dmax	0.8277	0.005	183.425	0.000	0.819	0.837
tmin	-0.3372	0.005	-74.208	0.000	-0.346	-0.328
atmmax	0.0373	0.005	7.866	0.000	0.028	0.047
wdsp(m/s)	0.0996	0.006	15.947	0.000	0.087	0.112
wgust(m/s)	-0.1289	0.007	-19.236	0.000	-0.142	-0.116

Omnibus:	3038.109	Durbin-Watson:	0.471
Prob(Omnibus):	0.000	Jarque-Bera (JB):	59165.785
Skew:	2.759	Prob(JB):	0.00
Kurtosis:	19.442	Cond. No.	4.44

Figure 23 Regression Model Summary

95% of variance in this dataset is explained by the features(prcp(mm) precipitation), radi(KJ/m2) is Solar Radiation, wdir(deg) is wind direction, dmax is (maximum dew point), tmin is minimum temperature, atmmax is maximum atmospheric pressure in (mb), wdsp(m/s) is wind speed and wgust(m/s) is wind gust. Wind gust refers to a brief increase in wind speed above the prevailing or average wind speed at a particular location

Here we can see that the relative humidity is depending on the precipitaion, solar radiation, wind direction, atmospheric pressure, temperature and sudden increase or decrease in wind speed.

Hypothesis Test –

F Test

```
H0: The fit of the intercept-only model and your model are equal.
H1: The fit of the intercept-only model is significantly reduced compared to your model.
```

Figure 24 Hypothesis F Test

```
F-statistic: 11364.742117489219, p-value: 0.0
```

Figure 25 F Test

From the F Test we can see this model is better than the intercept only model.
P value less than 0 so we can reject our null hypothesis and this this model performs better than null model.

T Test

```
H0: Coefficient is 0
H1: Coefficient is not 0
```

Figure 26 F Test Hypothesis

```
========== T Test ==========
H0: Coefficient is 0
H1: Coefficient is not 0
          Coef  T-Test  pvalues
0        const    0.00     1.00
1     prcp(mm)   17.79     0.00
2  radi(KJ/m2)  -18.73     0.00
3    wdir(deg)    1.98     0.05
4         dmax  183.42     0.00
5         tmin  -74.21     0.00
6       atmmax    7.87     0.00
7    wdsp(m/s)   15.95     0.00
8   wgust(m/s)  -19.24     0.00
```

Figure 27 F test

From the T Test we can see the coefficients are significant since p value is < 0.

AIC, BIC, RMSE, R-square, Adjusted R-square

Our adjusted R2 and R2 are similar so the model has learned the train data well. When I tried to remove any other feature, there was a decrease in my R2 and adjusted R2 so I will stick with this model.

As we can notice here, that AIC and BIC values are pretty low which is a good thing since AIC and BIC take into account the model's likelihood of fitting the data and penalize more complex models that may be overfitting the data. Therefore, lower AIC and BIC values suggest that a model provides a better balance between goodness of fit and complexity.

MSE

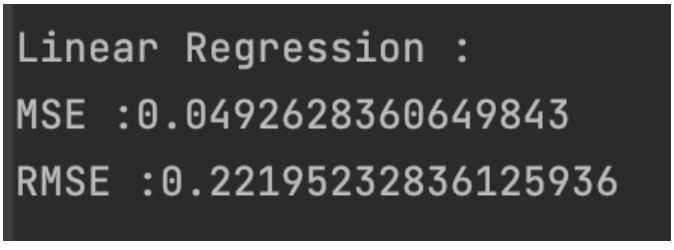


Figure 28 Linear Regression MSE

ACF of residuals

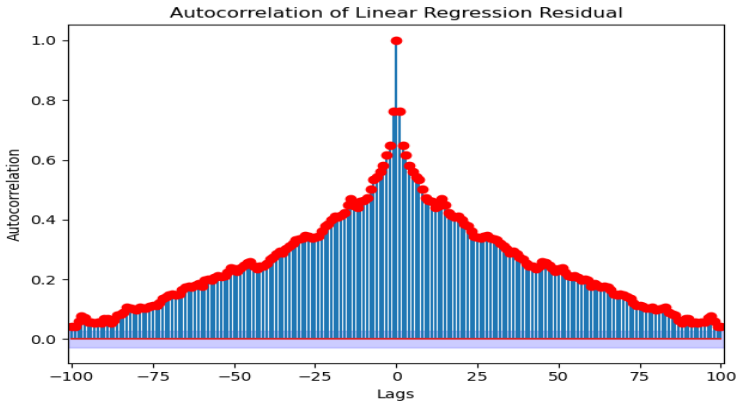


Figure 29 ACF of Residual from Linear Regression

Q value

```
Q value :41608.37
```

Figure 30 Q value for Linear Regression

The model is not generalized properly. High value.

Variance and mean of residual

```
Variance of residual : 0.05
Mean of residual : 0.00
```

Figure 31 Variance and Mean of Residual for Residual

ARMA

Preliminary model development procedure

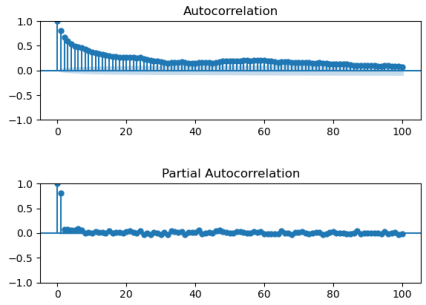


Figure 32 ACF/PACF Seasonally Differenced (365) data

From the ACF/PACF we can see a tail off in ACF, and cut off at 1 for pacf. This could be an ARMA(1,0).

Looking into GPAC

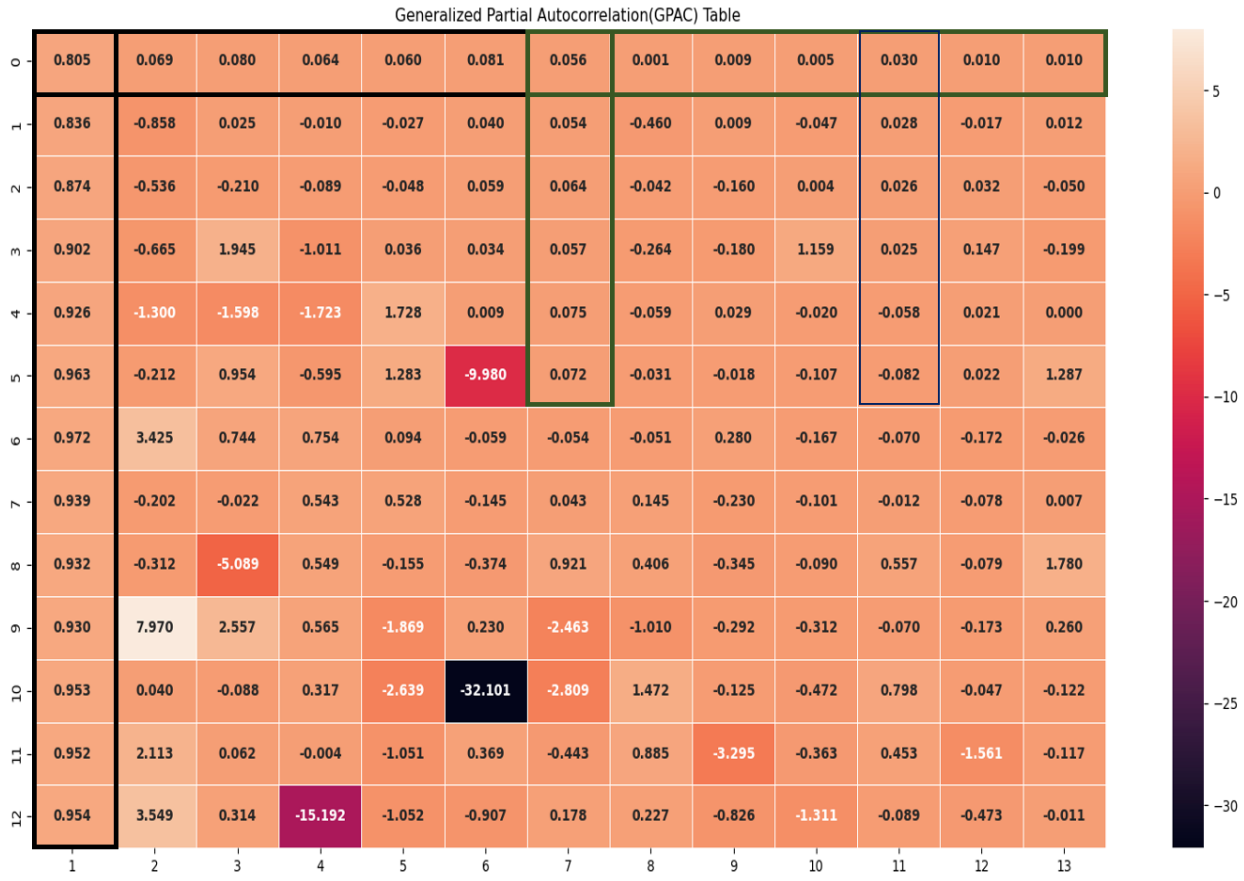


Figure 33 GPAC for Seasonally Differenced data

From the GPAC we can see a potential order of (1,0), (7,0) and (11,0).

Checking (1,0)

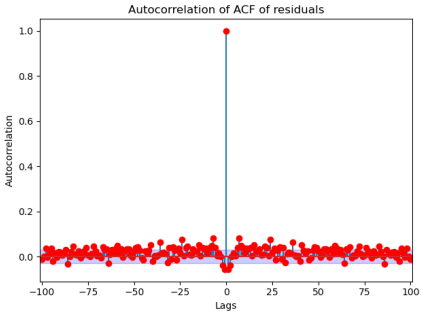


Figure 34 ACF Residual with order (1,0)

```
Q is 379.8133156558256 and chi critical is 134.64161685578915
The residual is NOT white
```

Figure 35 Chi Square Test with initial order(1,0)

Residual is not white.
So we will pass this residual again to the GPAC, and try to find the order. So that we can simply add that order to capture the remaining correlations left in the dataset.

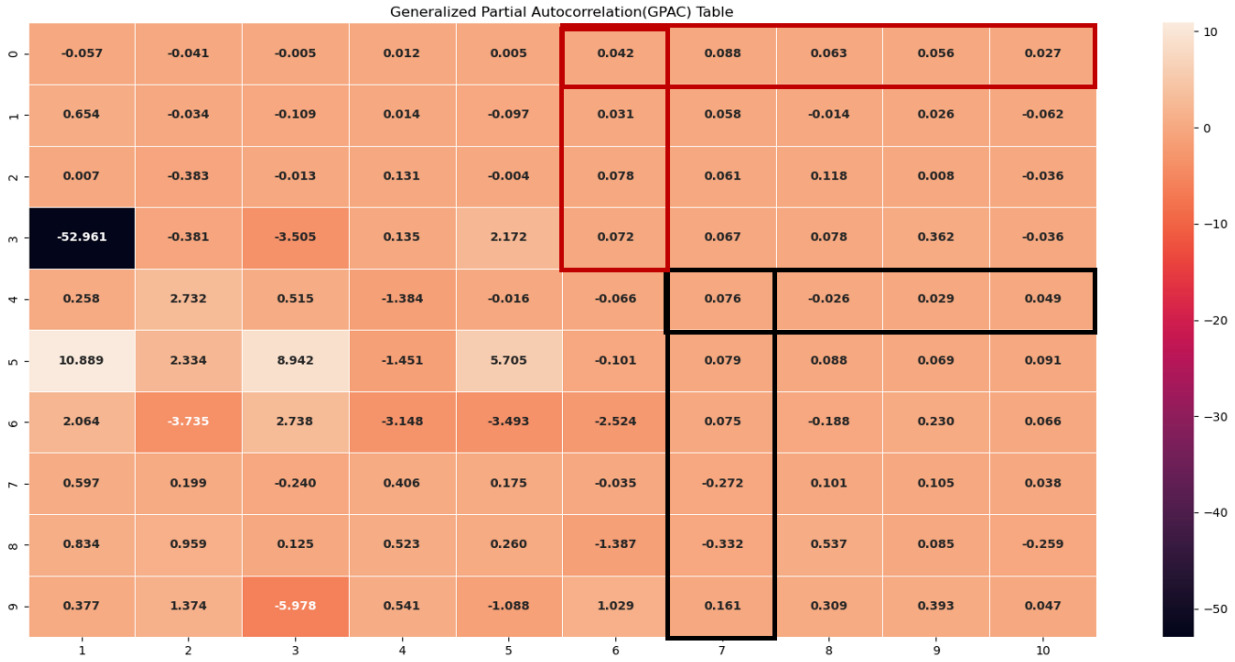


Figure 36 GPAC of Residual with order (1,0)

Here we can see an order of (6,0) and (7,4).
Adding this to our original order((1,0) + (6,0)) we have (7,0) which is the second order that we found in our original GPAC

Checking (7,0)

After taking order as **(7,0)** we are getting white noise as residual.

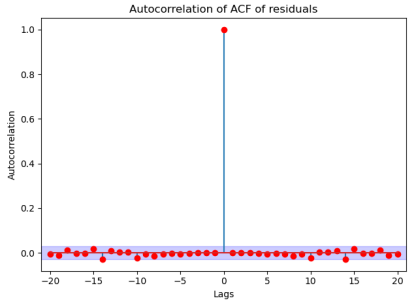


Figure 37 ACF of Residual with order (7,0) - Final Model

```
Q is 10.03692619783205 and chi critical is 27.68824961045705
The residual is white
```

From the Q statistics also, we can see that the residual is white which means the model has captured the information well.

```
==============================================================================
Dep. Variable:                hmdy(%)   No. Observations:                 4356
Model:                 ARIMA(7, 0, 0)   Log Likelihood               -2879.847
Date:                Tue, 09 May 2023   AIC                           5775.695
Time:                        18:32:14   BIC                           5826.729
Sample:                  03-04-2006   HQIC                          5793.707
                       - 02-04-2018
Covariance Type:                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7259      0.012     59.321      0.000       0.702       0.750
ar.L2          0.0060      0.015      0.388      0.698      -0.024       0.036
ar.L3          0.0273      0.016      1.745      0.081      -0.003       0.058
ar.L4          0.0178      0.016      1.137      0.256      -0.013       0.049
ar.L5       3.258e-05      0.015      0.002      0.998      -0.030       0.030
ar.L6          0.0406      0.017      2.454      0.014       0.008       0.073
ar.L7          0.0565      0.013      4.399      0.000       0.031       0.082
sigma2         0.2196      0.003     68.689      0.000       0.213       0.226
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              1176.25
Prob(Q):                              0.99   Prob(JB):                         0.00
Heteroskedasticity (H):               1.18   Skew:                            -0.05
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.54
===================================================================================
```

Figure 38 ARMA Model (7,0)

This is the final model that we have using order as (7,0)

Estimating ARMA parameters using Levenberg Marquardt algorithm

```
 theta_new
ay([[-7.26121458e-01],
    [-5.68501957e-03],
    [-2.75232348e-02],
    [-1.77082889e-02],
    [ 1.08953031e-04],
    [-4.11149949e-02],
    [-5.63822187e-02]])
```

Figure 39 Coefficients from LM algorithm

```
Esimated coefficient for order(7,0)
[[-7.26121458e-01]
 [-5.68501956e-03]
 [-2.75232348e-02]
 [-1.77082889e-02]
 [ 1.08953029e-04]
 [-4.11149949e-02]
 [-5.63822187e-02]]
-0.7564007127203285 < a1 < -0.6958422028487874
-0.043111691129700695 < a2 < 0.031741652018275134
-0.06495861757762092 < a3 < 0.009912147926313574
-0.055150378317269946 < a4 < 0.01973380045529649
-0.03732840333970217 < a5 < 0.03754630939821261
-0.07856265281916652 < a6 < -0.003667337050023041
-0.08668915163601228 < a7 < -0.026075285789347216
```

Figure 40 Parameter Estimates and Confidence Interval

```
Standard deviation : [[49.10719491]]
```

Figure 41 Standard deviation of parameter estimates

Here we can observe that the following coefficient have 0 in there confidence interval of so they would become insignificant a2, a3, a4 and a5.

Using the above information we have the following forecast function

Forecast Function

$$y(t) - 0.73y(t-1) - 0.041y(t-6) - 0.056y(t-7) = e(t)$$

Equation 9 Model Equation ARMA

Using this forecast function we can perform h step prediction.

The equations would be

$$y(t+1) = 0.73y(t) + 0.041y(t-5) + 0.056y(t-6) + e(t)$$

$$y(t+2) = 0.73y(t+1) + 0.041y(t-4) + 0.056y(t-5) + e(t)$$

$$y(t+3) = 0.73y(t+2) + 0.041y(t-3) + 0.056y(t-4) + e(t)$$

$$y(t+4) = 0.73y(t+3) + 0.041y(t-2) + 0.056y(t-3) + e(t)$$

$$y(t+5) = 0.73y(t+4) + 0.041y(t-1) + 0.056y(t-2) + e(t)$$

$$y(t+6) = 0.73y(t+5) + 0.041y(t) + 0.056y(t-1) + e(t)$$

$$y(t+7) = 0.73y(t+6) + 0.041y(t+1) + 0.056y(t) = e(t)$$

$$y(h) = 0.73y(h-1) + 0.041y(h-6) + 0.056y(h-7) = e(t) \text{ for } h = 8 \text{ and above.}$$

Equation 10 Forecast Equations

```python
def forecastFunction(y,test,train):
    """

    :param y: Train set
    :param test: test set (Relative Humidity)
    :param train: Train set (Relative Humidity)
    :return:
    """
    T = len(y) - 1
    test_len = len(test)
    y_hat = [0]
    y_hat.append(0.73 * y[T] + 0.041 * y[T - 5] + 0.056 * y[T-6])   # 1st step
    y_hat.append(0.73 * y_hat[1] + 0.041 * y[T - 4] + 0.056 * y[T-5])  # 2nd step
    y_hat.append(0.73 * y_hat[2] + 0.041 * y[T - 3] + 0.056 * y[T-4])  # 3rd step
    y_hat.append(0.73 * y_hat[3] + 0.041 * y[T - 2] + 0.056 * y[T-3])  # 4th step
    y_hat.append(0.73 * y_hat[4] + 0.041 * y[T - 1] + 0.056 * y[T-2])  # 5th step
    y_hat.append(0.73 * y_hat[5] + 0.041 * y[T - 0] + 0.056 * y[T-1])  # 6step
    y_hat.append(0.73 * y_hat[6] + 0.041 * y_hat[1] + 0.056 * y[T])  # 7th step
    for h in range(8, test_len+1):
        y_hat.append(0.73 * y_hat[h-1] + 0.041 * y_hat[h-6] + 0.056 * y_hat[h-7])  # 8 - last step predict

    y_hat = y_hat[1:]
    forecasted = pd.Series(y_hat)
    forecasted.index = test.index
    reverseTransform(forecasted, train, test)
```

Code 2 Forecast Function

```python
def reverseTransform(prediction,y_train,y_test):

    y_reversed = []
    I = len(y_train)
    seasonality = 365
    for i in range(0,len(y_test)):
        if i <seasonality:
            y_reversed.append(prediction[i] + y_train[-seasonality+i])
        else:
            k = i-seasonality
            y_reversed.append(prediction[i] + y_reversed[k])
    forecasted_values = pd.Series(y_reversed)
    forecasted_values.index = prediction.index

    forecasted_values.plot(label='Forecast')
    y_test.plot(label='Actual Test Data')
    plt.title('Predictions Versus Test')
    plt.legend(loc='upper right')
    plt.grid()
    plt.tight_layout()
    plt.show()

    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')
    plt.title('Predictions')
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.show()
```

Code 3 Reverse Transformation

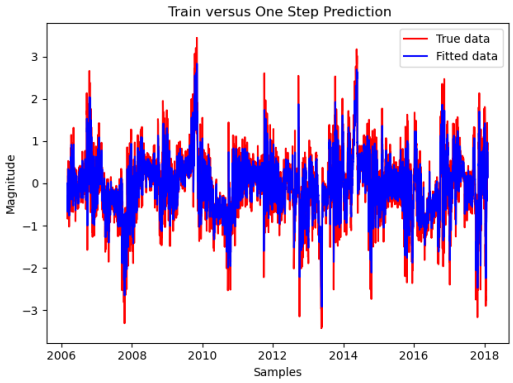


Figure 42 One Step on ARMA(7,0)

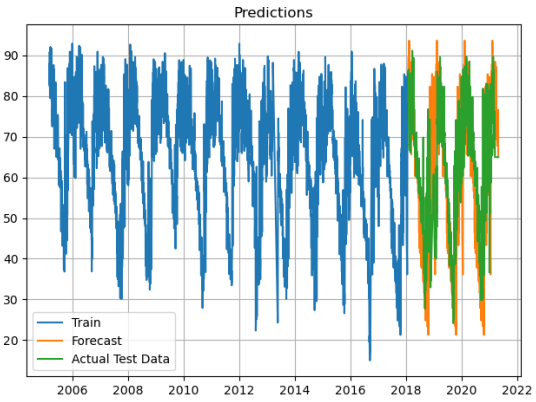


Figure 43 h step Prediction on order (7,0)

```
ARMA Model(7,0)
Mean Square error forecasted  192.13
Root Mean Square error forecasted 13.86
```

Figure 44 Forecasted Error ARMA(7,0)

Since the model equation and prediction was on differenced data, I have reversed transformed my dataset to get original values.

Diagnostic Analysis

Estimated Coefficients and Confidence Interval

```
Esimated coefficient for order(7,0)
[[-7.26121458e-01]
 [-5.68501956e-03]
 [-2.75232348e-02]
 [-1.77082889e-02]
 [ 1.08953029e-04]
 [-4.11149949e-02]
 [-5.63822187e-02]]
-0.7564007127203285 < a1 < -0.6958422028487874
-0.043111691129700695 < a2 < 0.031741652018275134
-0.06495861757762092 < a3 < 0.009912147926313574
-0.055150378317269946 < a4 < 0.01973380045529649
-0.03732840333970217 < a5 < 0.03754630939821261
-0.07856265281916652 < a6 < -0.003667337050023041
-0.08668915163601228 < a7 < -0.026075285789347216
```

Figure 45 Estimated Coefficients and Confidence Interval

Roots of Polynomial

```
Roots of the polynomial are :
num : []
den : [ 0.93  0.51  0.51 -0.11 -0.11 -0.5  -0.5 ]
```

Figure 46 Roots of polynomial

There is no zero pole cancellation.

Covariance Matrix

```
[ 0.0002292083198634685 -0.0001674928091687683 -1.3125144599343014e-06 -6.547957375241559e-06 -4.425269433481064e-06 -5.409226041031765e-08 -1.8866173006929582e-05 ]
[ -0.0001674928091687683 0.0003501889362767887 -0.0001666898590245767 3.076081990059327e-06 -3.8110347757083517e-06 -4.338971766565305e-06 -1.2925932005441038e-07 ]
[ -1.3125144599344493e-06 -0.0001666898590245767 0.0003503519704657173 -0.0001666036035978577 3 0.056167025941441 3e-06 -3.988819605552041e-06 -4.241069421905311e-06 ]
[ -6.5479573752413576-06 3.0760819900590684e-06 -0.0001666036035978576 3 0.0003504775144026055 5 -0.0001666112824401033 6 2.9726712422883216e-06 -6.5204552773747155e-06 ]
[ -4.425269433481191e-06 -3.8110347757081768e-06 3.0561670259412935e-06 -0.0001666112824401033 6 0.0003503889129740786 -0.0001666040058275685 6 -1.4491295276598843e-06 ]
[ -5.4092260410297653e-08 -4.338971766565336e-06 -3.988819605552014e-06 2.9726712422883466e-06 -0.0001666040058275686 4 0.0003505817702599 82 -0.0001677542640195170 4 ]
[ -1.8866173006929576-05 -1.2925932005443407e-07 -4.241069421905274e-06 -6.520455277374722e-06 -1.4491295276598972e-06 -0.0001677542640195170 7 0.0002962754580484432 ]
```

Figure 47 Covariance Matrix

Since the residual is white the model has captured all information and is unbiased.

I tried to use SARIMA with the order of (1,365) for my dataset but due to computational requirements I was not able to use it. For all other models ARMA(7,0) performed best for my dataset.

Deep Learning

```
model = Sequential()
model.add(LSTM(64, activation="relu", input_shape=(train_X.shape[1], train_X.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(64, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.summary()
history = model.fit(train_X, train_y, batch_size=16, validation_split=.1, epochs=10, verbose=1)
plt.figure()
plt.plot(history.history['loss'], 'r', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.legend()
plt.show()
dataset = df2.values
test_data = scaled_data[train_size - 365:, :]
x_test = []
y_test = dataset[train_size:, -1]
```

Code 4 LSTM Code

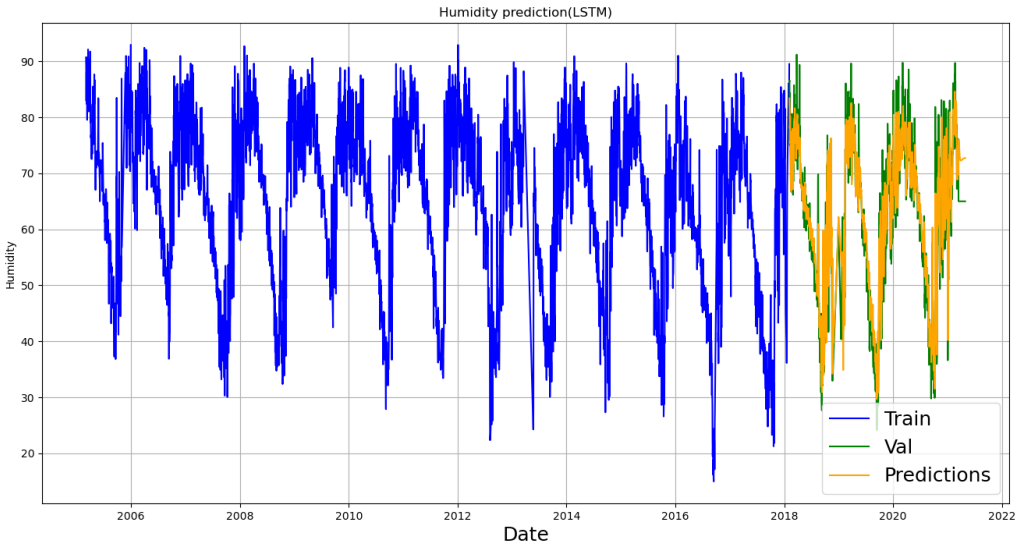


Figure 48 LSTM Predictions

```
Mean Square Error for LSTM :  27.14
Root Mean Square Error for LSTM :  5.21
```

Figure 49 Mean Square Error from LSTM

Final Model Selection

Final model would be LSTM and ARMA model since they have the lowest MSE.
From the traditional model ARMA Is performing the best
Overall LSTM is performing best.

My best performing model is :

```
ARMA Model(7,0)
Mean Square error forecasted  192.13
Root Mean Square error forecasted 13.86
```

```
Mean Square Error for LSTM :  27.14
Root Mean Square Error for LSTM :  5.21
```

```
mean square error for Holt Winter is : 361.21
Root Mean Square Error Holt Winter is : 19.00
```

As we can see the best performing model is LSTM and then it's the ARMA Model.
As residual in ARMA model is white and the mean square error is also low, as compared to other models I would pick ARMA Model.

Summary and Conclusion

- Dataset is highly seasonal.
- Holt Winter performs well on seasonal dataset.
- LSTM outperforms other models.
- Best traditional model is ARMA(7,0).

Limitations

- ACF/PACF: The ACF and PACF analysis suggests an order of (1,365) for the best model representation based on the raw dataset's autocorrelation and partial autocorrelation patterns.
- SARIMA Limitation: Running a SARIMA model with such high seasonality (365) is not feasible due to computational constraints or other limitations.
- Interpolation and Errors: Due to the presence of missing values, interpolation has been used to fill in the gaps. However, interpolation can introduce some errors that might affect the accuracy of the predictions.
- Time-dependent Independent Variables: In linear regression, when the independent variables are also time-dependent, making predictions beyond one step ahead becomes challenging due to assumptions about known future values of the independent variables.

References

- Hyndman, R.J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd edition). OTexts. https://otexts.com/fpp2/holt-winters.html#
- statsmodels.regression.linear_model.OLS - statsmodels 0.15.0 (+6). (n.d.). https://www.statsmodels.org/devel/generated/statsmodels.regression.linear_model.OLS.html