WEATHER FORECASTING USING ARIMA MODEL

MONISH KUMAR RAMBA

ABSTRACT

The advances made in the field of weather forecasting have shaped the economy as it is today. From the field of agriculture, where knowledge of weather has helped improve harvest by multiple times, to the field of transportation and travel, or scheduling of any event, or preparing for harsh weather conditions, weather forecasting has made its mark. While complex NWP models are currently employed for forecasting weather, the inclusion of machine learning techniques can help optimize the output and also significantly fasten the process.

The aim of this project is to analyze machine learning techniques that can be employed to identify weather patterns from historical data, identify the trends and seasonality among the data and then based on that make future predictions. ARIMA i.e. Auto-Regressive Integrated Moving Average is a machine learning model that studies time series, identifies its unique features such as stationarity, trends, seasonality, etc. Seasonal ARIMA or SARIMA is a version of ARIMA which recognises seasonality, i.e. repeating patterns in the time series. We will use a dataset of Delhi's hourly temperature conditions from 1997 - 2016 and apply data transformation techniques to make it suitable time series for ARIMA and SARIMA. We then build these models and evaluate the performance of the model.

INTRODUCTION

Weather Forecasting is very essential to a lot of daily tasks. Be it planning for day-to-day tasks, scheduling important events such as festivals, celebrations, etc or in the field of agriculture, it plays a pivotal role. As a matter of fact, we often undermine its importance owing to the advances made by the different meteorological departments in this field. The whole agricultural cycle, involving plantation of crops, irrigation and harvesting are shaped around the weather forecast. The advances made in this field have ensured minimal losses and maximum yield, directly impacting the economy.

It is not only the field of agriculture that works so closely with weather forecasting. Timely warnings of natural disasters have greatly helped in mitigating the loss of lives and property. Consider India, for example, a country prone to cyclones. Without a proper system to foretell incoming cyclones, the coastlines of India would have been long ravaged by now, not to mention the loss of lives and property which will subsequently follow. Transportation and Travel also rely on weather forecasting. Especially in the field of air travel, in addition to predicting unsafe weather for flight, it also helps in route optimization and scheduling adjustments. The wind speed over different regions, for example, helps predict the quickness/delay with which planes might arrive.

Of course, the primary concern of Weather Forecasting is to predict day-to-day weather for the daily needs. Also, it predicts when certain events like heatwaves, cold spells, or poor air quality, a specialty of Delhi, are to set in, giving the public enough time to brace for it. It also contributes to the field of Energy Production, especially solar and wind energy. Thus, it can be quite easily concurred that weather forecasting has a great impact on the economy.

Now that we have emphasized on the importance of Weather forecasting and how it shapes the global economy, let us delve into the scientific methods that have been used prevalently till now. The single term encompassing all these methods is meteorology. In simple terms, it is the science concerning the atmosphere and its phenomena. The prediction of weather and climate is simply one of the many disciples of meteorology. Meteorologists study atmospheric properties, satellite data and radar information and then apply this study to predict the weather.

Meteorology originated from Ancient Greek and Chinese civilizations, trying to decipher the relations between patterns and atmospheric phenomena. Renaissance periods witnessed the birth of the very first weather instruments, thermometer, barometer, and anemometer, thanks to the contributions of scientists Torricelli and Daniel Fahrenheit. Atmospheric properties such as temperature and pressure were introduced. Fast forward to a few centuries back, advances in chemistry such as Dalton's gas laws and Hailey's study of trade winds and monsoons helping scientists understand the atmosphere's composition and dynamics.

20th century was when the development of modern-day models started. With significant advances made in the physics and chemistry governing the weather patterns and atmospheric phenomena, complex mathematical equations were developed to explain the patterns and further forecast weather. Once the correct equations were established, numerical methods were applied to solve these equations, and this system is what came to be known as NWP or Numerical Weather Prediction.

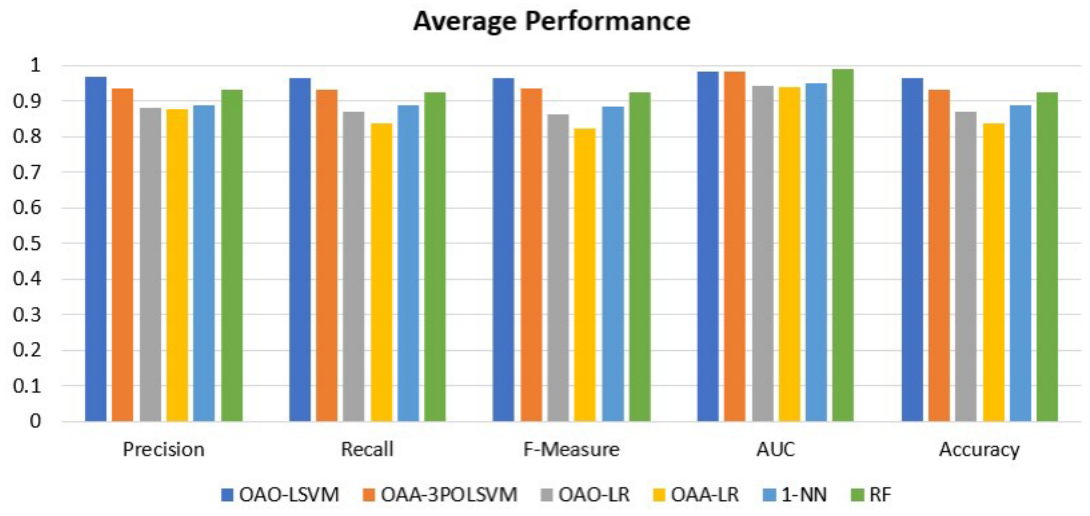
The launch of artificial satellites further broadened the scope, providing crucial weather data such as cloud cover and moisture distribution. The recent leaps made in the field of computing have allowed for the development of more sophisticated NWP models. These models not only predict with more accuracy but also predict other features such as humidity, rainfall.

Researchers have explored the possibility of using machine learning in this field, and the conclusion they came to is that while pure machine learning models cannot replace NWP models for weather prediction, they do act as very good complements for NWP models, because they can efficiently and effortlessly handle large chunks of data, recognize patterns better, and remove any sort of bias/error from the final NWP model output. They are also handy in predicting sudden weather events such as storms or cyclones. By using old data from such events, they can identify the precursor to it and thus help generate a timely warning.

But since it is not within the scope of this project to complement NWP models with Machine Learning techniques, we will be looking at approaches which attempt to forecast weather with a high accuracy by purely relying on Machine Learning techniques. The Literature Survey section below is a brief overview of a few such methods.

LITERATURE SURVEY

Weather prediction has become a popular field among researchers lately due to the multiple domains it covers. It requires a fusion of both science and technology to process, compute and design the forecasting model. In [1], linear regression and artificial neural networks are used. These two techniques give us a dataset consisting of average temperature, humidity, dew point, pressure, and wind speed. Then for probability calculation, supervised learning is used. The original dataset is partitioned into small datasets of size 10x5 each, and then the decision tree algorithm is applied on each of these datasets. The probability is compiled on resulted outcome, and the



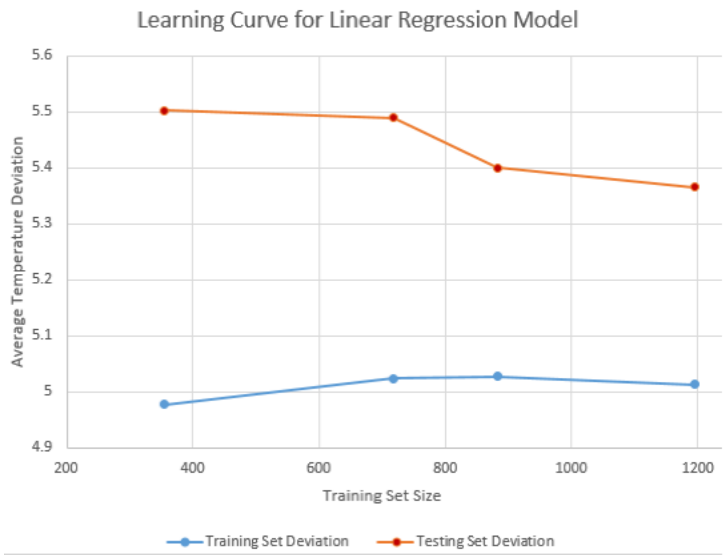
confusion matrix generated is used to obtain the performance metrics.

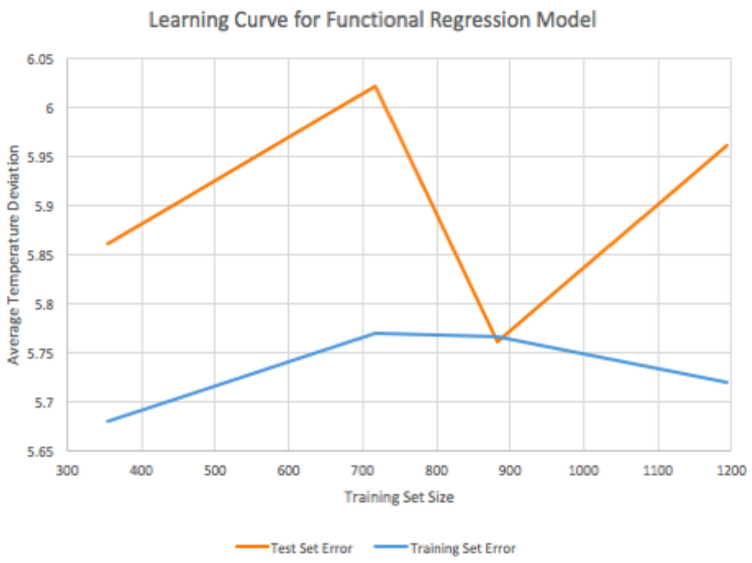
A multi-classification methodology using Machine Learning is followed in [2]. The dataset used consists of 5 binary classified data – Rain, Drizzle, Sun, Snow, and Fog. In addition, max temp, min temp, precipitation and wind speed are also taken as parameters. 4 different algorithms are applied, in which 2 required classification. One Against One (OAO) and One Against All (OAA) multi-classification methods were applied to split the single multi-class problem into 5 binary class sub-problems. This classification simplifies the task for Support Vector Machines (SVM) and Linear Regression. While the other 2 algorithms, k-Nearest Neighbors (kNN) and Random Forest Classification didn't require any. The summary of the results is:

- Linear SVM with OAO Classification – Accuracy 96.64%

- 3rd degree polynomial SVM with OAA Classification – Accuracy 93.08%
- Linear Regression with OAO Classification – Accuracy 86.92%
- Linear Regression with OAA Classification – Accuracy 83.69%
- kNN (k=1) – Accuracy 89.04%
- Random Forest Classifiers – Accuracy 92.60%

Paper [3] applies simple machine learning techniques to design an accurate weather prediction classification model. BMP280 (Pressure-Humidity sensor), DHT22 (Temperature-Humidity sensor) and ESP8266 (Microcontroller) are used to measure temperature, humidity, pressure, and rainfall and then the average of these is predicted over the span of next 28 days. The Machine Learning techniques used in [4] are also Naïve Bayes and decision tree classifiers. Once the dataset has been preprocessed to adjust null values, crucial features are extracted and finally the model is trained. A web interface was also built so for the user to enter the location details and date/time.





Published by Stanford University, [5] used linear and functional regression. While the linear regression model predicted the maximum and minimum temperature, the functional regression model could predict more variables. Learning curves for both these models were also generated to give us a good estimate as to how dependent the model was on the training dataset size. While the professional forecasting methods outperformed both these models, the discrepancies significantly dropped for the later days, indicating that the models might be more suitable for long term weather prediction.

Both Data Mining and Machine Learning Techniques are being used in [6]. Support Vector Regression (SVR) and Artificial Neural Networks (ANN) are used. SVR is like SVM, except SVM does classification-based prediction while SVR uses regression-based prediction. ANN is based on a large collection of mathematical neurons simulating the biological brain and how it solves problems. Then to evaluate these models, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are calculated. Rainfall prediction was done using both these models using two types of datasets:

- A historical dataset of only rainfall
- A dataset with both rainfall and temperature

The SVR outperformed ANN with 0.95% and 0.17% error rate in both single and combined dataset. While for temperature prediction, the ANN outperformed SVR.

In [7], the objective is to achieve higher accuracy for weather forecast using machine learning. Decision Tree, Clustering and Logistic Regression are the techniques used here. [8] Provided an overview on the use of Machine Learning techniques in this field. For Weather Forecasting, mainly Bayesian Networks and Neural Networks are used, while for Data Mining for Weather Forecasting, Classification, Clustering, and Decision Tree are used. Recurrent Network and Convolution Networks are some of the Deep Learning Techniques used. It also discussed the some of the challenges faced by these methods such as incomplete data.

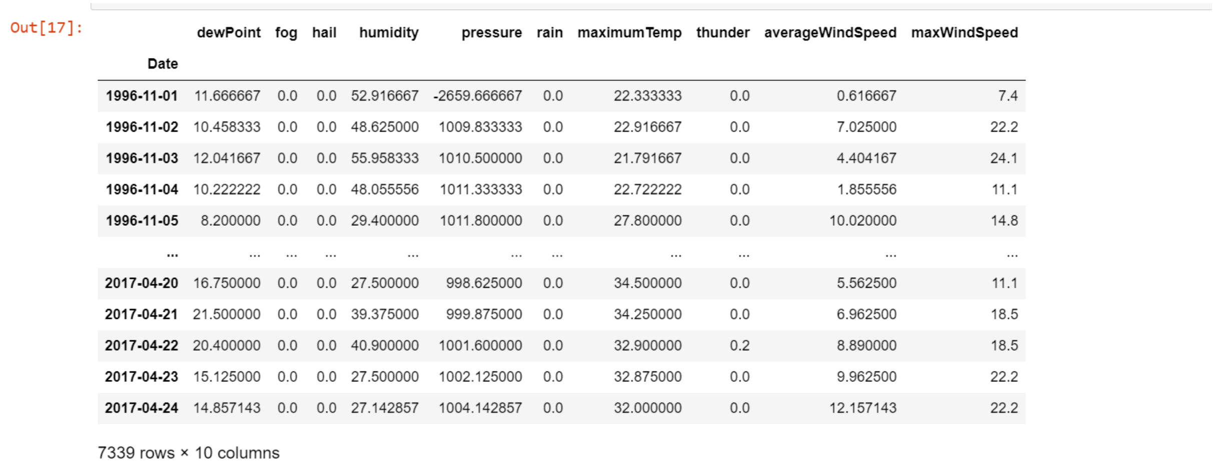
In [9], Linear Regression, Isotonic Regression, Polynomial Regression of degree 2 and 3, and SVR are used to forecast the weather in Bangladesh. Its further predicted for three different season – Winter, Summer and Rainy, while its also predicted yearly. After calculating the Mean Square Error, Mean Absolute Error, Median Absolute Error, and R2_score, it was found that Isotonic regression performed the best on yearly data as well as through all 3 seasons. However, the isotonic model was observed to perform poorly on the test data. Thus, the authors recommended polynomial regression or SVR of higher degrees for annual or seasonal temperature prediction.

Adaptive Network Fuzzy Interface System (ANFIS) and Auto Regressive Integrated Moving Average (ARIMA) models are used for weather prediction in Istanbul in [10]. The specialty of ANFIS in addition to other ML models out there is that it makes use of both neural networks and fuzzy logic systems. ARIMA, on the other hand, is a statistical tool used for understanding and predicting future trends in time series' data. With a train-test split of 95:5, the outcome of both the models were observed based on Mean Absolute Error, Root Mean Square Error, and R2 Score. It was found that the ANFIS model performed better than the ARIMA model.

PROPOSED METHODOLOGY

DATA RETREIVAL AND CLEANING

The dataset used for training and testing the model is picked from Kaggle[11]. It consists of a detailed weather survey recorded hourly of New Delhi from 1996 to 2017. Since it has been recorded hourly for 20 years, there is bound to be a lot of missing data. There are 19 different fields apart from the datetime, which are wind speed, wind direction, weather condition (Smoky, Hazy, etc.), dew point, fog, hail, heat index (almost entirely null), humidity, precipitation (again almost entirely null), pressure, rain, snow, maximum temperature, thunder, tornado, etc. First, the useless fields and the fields which were mostly null were dropped. The null values in wind speed column were filled with 0. Thus, the dataset was eliminated to only 9 fields – dew point, fog, hail, humidity, pressure, rain, maximum Temperature, thunder, and average wind speed. Finally, the data was converted to daily time series by taking the mean for all the fields with the hourly data.



Out[17]:

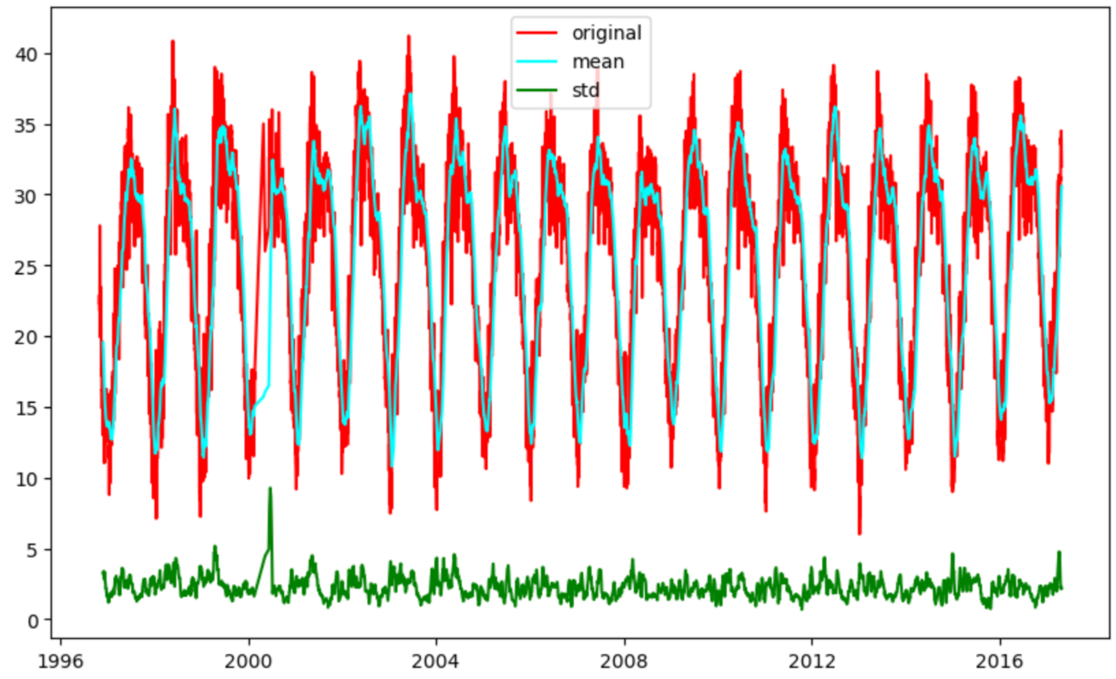
Date	dewPoint	fog	hail	humidity	pressure	rain	maximumTemp	thunder	averageWindSpeed	maxWindSpeed
1996-11-01	11.666667	0.0	0.0	52.916667	-2659.666667	0.0	22.333333	0.0	0.616667	7.4
1996-11-02	10.458333	0.0	0.0	48.625000	1009.833333	0.0	22.916667	0.0	7.025000	22.2
1996-11-03	12.041667	0.0	0.0	55.958333	1010.500000	0.0	21.791667	0.0	4.404167	24.1
1996-11-04	10.222222	0.0	0.0	48.055556	1011.333333	0.0	22.722222	0.0	1.855556	11.1
1996-11-05	8.200000	0.0	0.0	29.400000	1011.800000	0.0	27.800000	0.0	10.020000	14.8
...	...	...	...	...	...	...	...	...	...	...
2017-04-20	16.750000	0.0	0.0	27.500000	998.625000	0.0	34.500000	0.0	5.562500	11.1
2017-04-21	21.500000	0.0	0.0	39.375000	999.875000	0.0	34.250000	0.0	6.962500	18.5
2017-04-22	20.400000	0.0	0.0	40.900000	1001.600000	0.0	32.900000	0.2	8.890000	18.5
2017-04-23	15.125000	0.0	0.0	27.500000	1002.125000	0.0	32.875000	0.0	9.962500	22.2
2017-04-24	14.857143	0.0	0.0	27.142857	1004.142857	0.0	32.000000	0.0	12.157143	22.2

7339 rows × 10 columns

Time-series after all pre-processing

DATA ANALYSIS AND TESTS FOR STATIONARITY

Now we move on to analyze the data. ARIMA model, unlike standard models such as Linear Regression or SVM does not rely on other features. Instead, it uses time series modelling and makes use of features such as moving averages to predict. Thus, the other fields in our case are useless if we are trying to predict a single field. Let us take the case of maximum temperature first. First, we calculate the rolling mean and the rolling standard deviation of the dataset with only maximum temperature. Plotting the original dataset, its rolling mean and rolling standard deviation, we obtain the following graph:

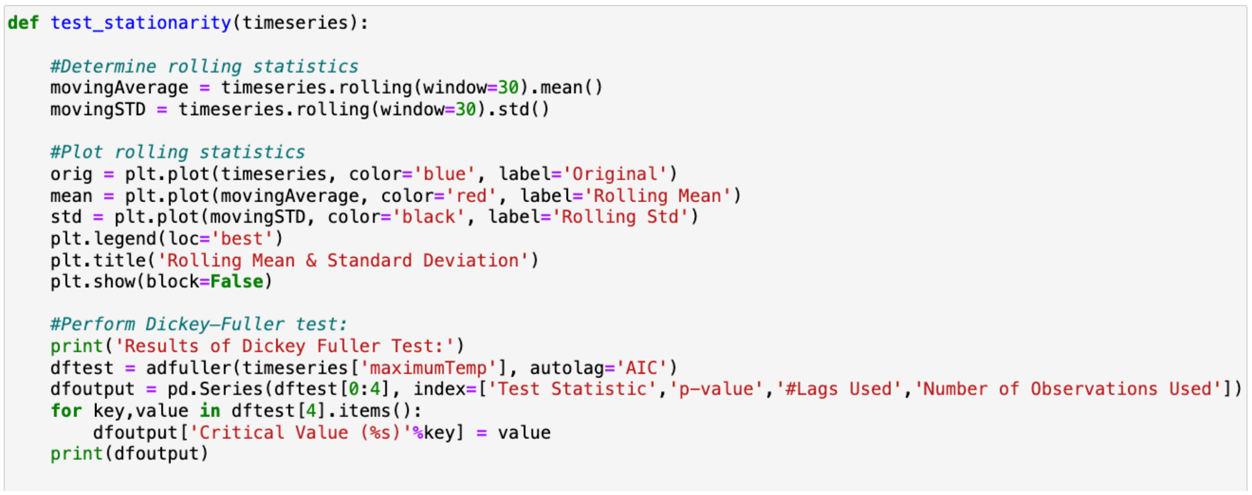


Original dataset and its rolling mean and standard deviation

The key method to measuring the stationarity of a series is by using the Augmented Dickey-Fuller (ADF) test. To transform our dataset into a stationary one, we can either use transformations, differencing or a combination of both. There are three different types of transformation which can be applied - **log transformation**, **exponential decay transformation,** and **time-shift transformation**. Then we can perform differentiation by subtracting the original series with its rolling mean. Make sure to note the *order of differencing 'd'* here as it is one of the crucial parameters for defining both the SARIMA and ARIMA model.

The test_stationarity is the function determining the stationarity. First the rolling mean, rolling standard deviation and the original dataset are plotted. The aim is to achieve a close to flat rolling mean or rolling standard deviation curve. Then the results of ADF-test are noted, and the following two conditions must be met:

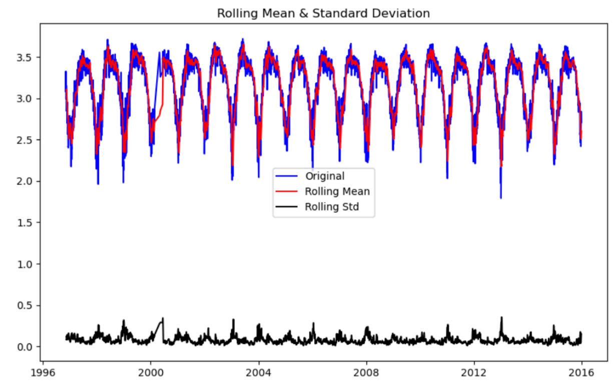
- P-value should be less than 0.05
- The test statistic should be less than all 3 critical values, or preferably it should be a highly negative value.



```python
def test_stationarity(timeseries):

    #Determine rolling statistics
    movingAverage = timeseries.rolling(window=30).mean()
    movingSTD = timeseries.rolling(window=30).std()

    #Plot rolling statistics
    orig = plt.plot(timeseries, color='blue', label='Original')
    mean = plt.plot(movingAverage, color='red', label='Rolling Mean')
    std = plt.plot(movingSTD, color='black', label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey–Fuller test:
    print('Results of Dickey Fuller Test:')
    dftest = adfuller(timeseries['maximumTemp'], autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
```

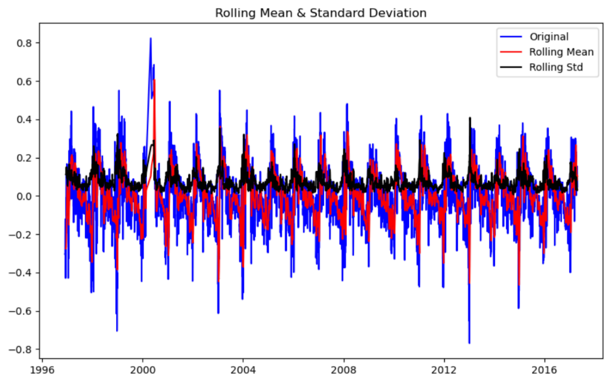
Function to test stationarity of time series

First we test apply log transformation on the original dataset and test its stationarity. While the rolling mean and standard deviation still have their characteristic curves, the ADF test gives satisfactory results, with both conditions being followed. Still, we apply first order differencing on this data by subtracting its rolling mean from it. We do achieve close-to-flat rolling mean and std curves but with a lot of noise. The ADF-tests are also better in this case. But for simplicity of reverting changes, we go ahead with the log transformed time series.



```
Results of Dickey Fuller Test:
Test Statistic                 -6.966539e+00
p-value                         8.885064e-10
#Lags Used                      3.500000e+01
Number of Observations Used     6.823000e+03
Critical Value (1%)            -3.431309e+00
Critical Value (5%)            -2.861964e+00
Critical Value (10%)           -2.566996e+00
dtype: float64
```

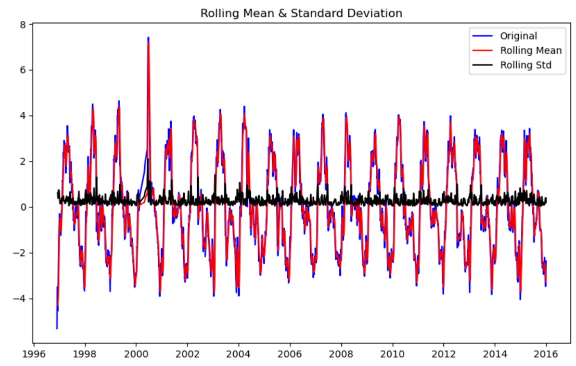
Stationarity for Log transformed data



```
Results of Dickey Fuller Test:
Test Statistic                 -8.185168e+00
p-value                         7.935558e-13
#Lags Used                      3.200000e+01
Number of Observations Used     7.277000e+03
Critical Value (1%)            -3.431249e+00
Critical Value (5%)            -2.861937e+00
Critical Value (10%)           -2.566981e+00
dtype: float64
```

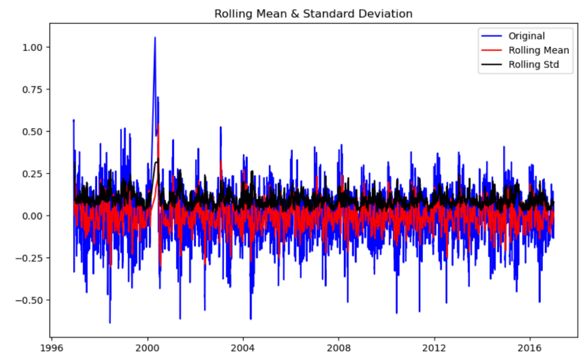
Stationarity for Log transformed then differenced data

Now we apply exponential decay transformation. This time the transformation is applied on already differenced data series. Both the ADF-tests and the plots obtained in this case are impressive, suggesting complete stationarity. Unfortunately, handling data like this will be complex when it comes to reverting the changes back. Furthermore we also apply a 2nd order of differencing on this exponential decay time series, after which showed less signs of stationarity, suggesting over-differencing.



```
Results of Dickey Fuller Test:
Test Statistic                -7.573688e+00
p-value                        2.797780e-11
#Lags Used                     7.000000e+00
Number of Observations Used    6.822000e+03
Critical Value (1%)           -3.431309e+00
Critical Value (5%)           -2.861964e+00
Critical Value (10%)          -2.566996e+00
dtype: float64
```

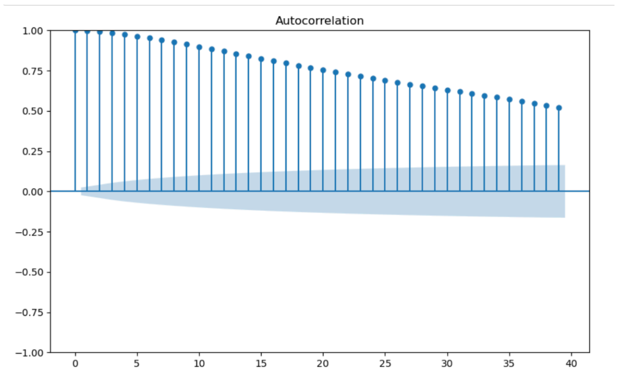
Stationarity for differenced exponentially decay transformed data



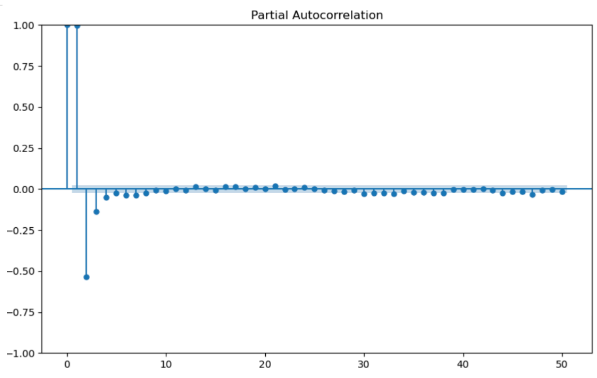
```
Results of Dickey Fuller Test:
Test Statistic                  -23.907348
p-value                           0.000000
#Lags Used                        5.000000
Number of Observations Used    7188.000000
Critical Value (1%)              -3.431260
Critical Value (5%)              -2.861942
Critical Value (10%)             -2.566984
dtype: float64
```

Stationarity for differenced —> exponential trans. —> 2nd order diff. Data

We decide to first go forward with the exponential decay transformed data. We won't be able to get the temperature predictions due to the complexity involved in reversing this transformation, but at least we can apply the same transformation on the original dataset and check for the root



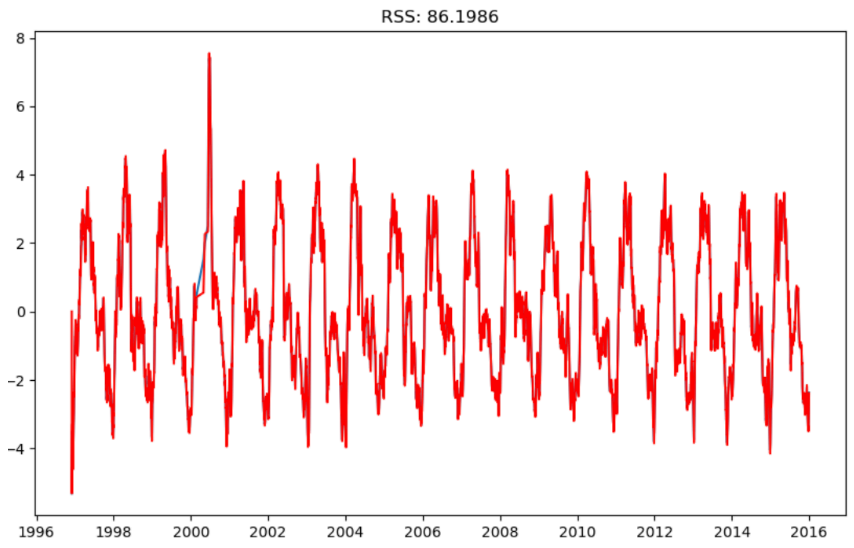
Auto-Correlation Factor Plot



Partial Auto-Correlation Factor Plot

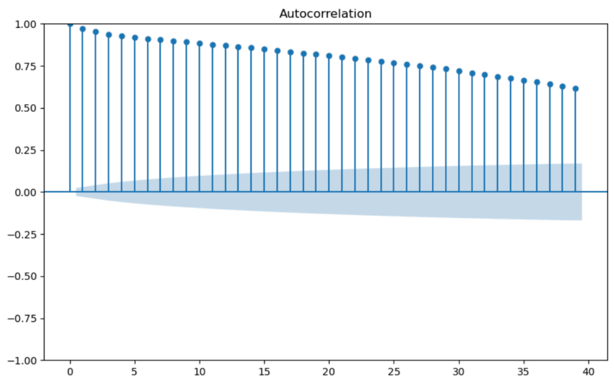
mean square error. We first check for the Auto-Correlation (ACF) and Partial Auto-Correlation (PACF) plots. They do not show any irregular spikes, suggesting we can proceed with the model

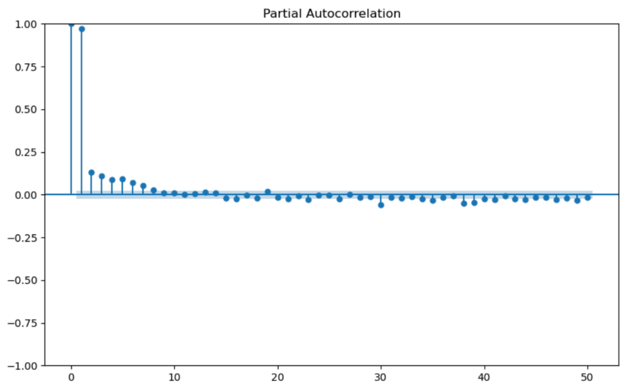
First we plot the Auto-Regressive And Moving Average Models before to get an idea of their RSS. We get RSS values of 86.2 and 92 respectively, for orders (2,1,0) and (0,1,2). After this, we integrate the AR and MA models to plot the ARIMA model. We get an RSS value even lower than both of those 86.1, suggesting that our model is performing nicely. However, this isn't a good enough metrics. So we apply exponential transformation on the original dataset, and then compare the predicted exponential values from 1996-2016 with the original exponential values. We finally get a **MSE of 0.0126**, hinting that our model performed quite well on the training data.



ARIMA Model with Exponential Decay time-series

Now, we move on the the log-transformed dataset. This time getting back the temperature predictions should not be difficult as the order of differencing here is 0, so no values are lost due to rolling mean. Only log transformation is applied. We check the ACF and PACF plots for this data. The PACF plots are slightly unreliable due to the fact that they spill out of the confidence band, but still let's go ahead with this data

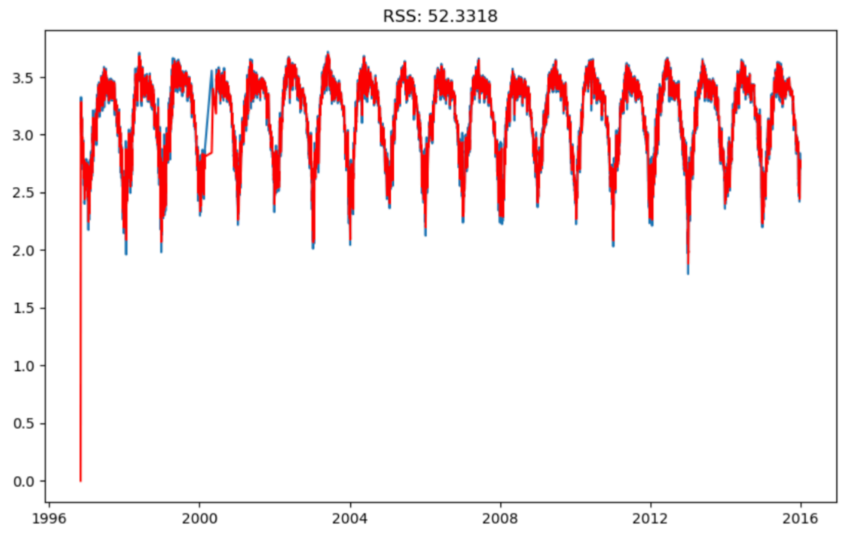




Auto-Correlation Factor Plot

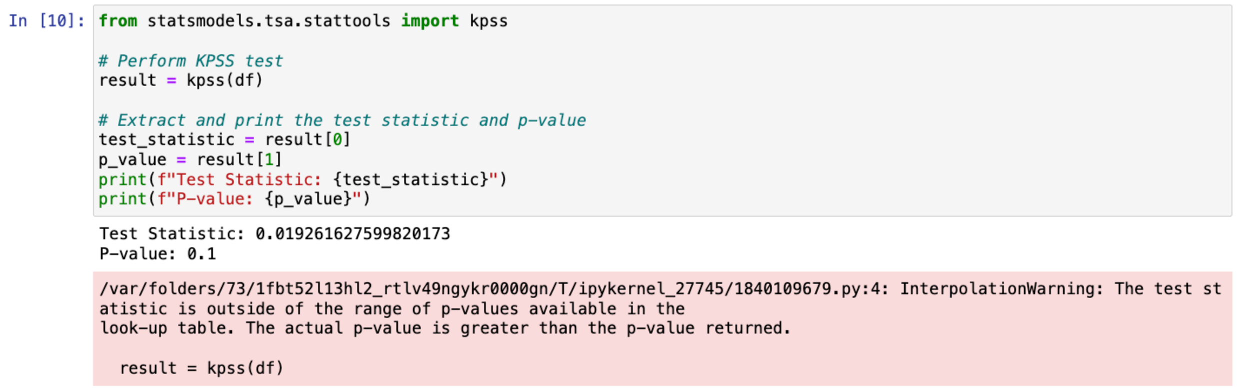
Partial Auto-Correlation Factor Plot

Again we follow the same process of first constructing the AR and MA models. We follow the same order and get RSS values of 53.5 and 52.9. This is better than the exponential dataset model because in general a low value of RSS means better performance. Now we combine these to get the ARIMA model, with an RSS value of 52.3. As for the predictions, we first get the predictions from the model and then we reverse transform it from log by taking exponential of these predictions. Then we compare this with the original maximum temperature dataset to get **a MSE of 2.9**. In other words, on an average, **the predicted maximum temperature is 2-3 degrees Celsius off from the actual maximum temperature.**



ARIMA Model with Log Transformed time-series

Furthermore, we also build a SARIMA, or in other words a Seasonal ARIMA model. A SARIMA model is better suited for understand repeating trends and curves. As the name suggests, it can identify seasonality within the time series quite efficiently. However for this series, we use a different dataset. In this case to simplify the time series, I have converted its frequency from daily to monthly. So, it has recorded the monthly average maximum temperature from 1996-2016. We perform two different tests - **ADF Test** and **KPSS Test** to ensure stationarity within the series. The ADF test yielded quite impressive results. Both the rolling mean and rolling standard deviation curves are almost flat and the main curve has a sinusoidal pattern without much noise. The p-value, 0.02, is well-below the set limit of 0.5. The only concern is that the test-statistic is not highly negative and a bit close to the critical values. However that shouldn't be much of a concern. As for the KPSS-test, we only had to ensure that p-value is greater than 0.5, which it is, 0.1 in our case.

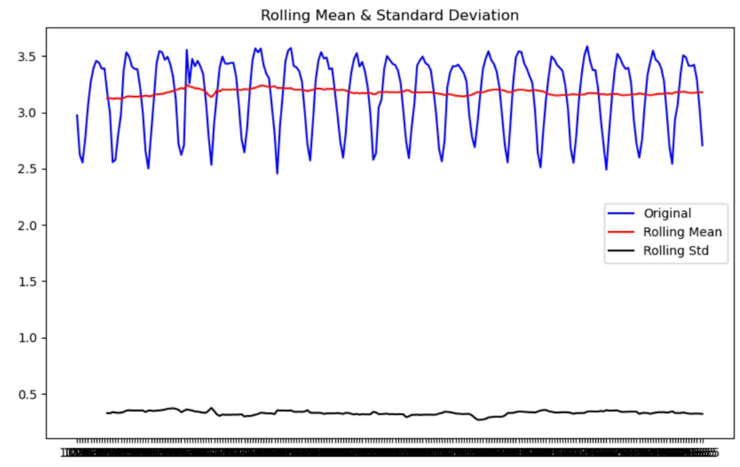


```
In [10]: from statsmodels.tsa.stattools import kpss

         # Perform KPSS test
         result = kpss(df)

         # Extract and print the test statistic and p-value
         test_statistic = result[0]
         p_value = result[1]
         print(f"Test Statistic: {test_statistic}")
         print(f"P-value: {p_value}")
```

```
Test Statistic: 0.019261627599820173
P-value: 0.1
```

```
/var/folders/73/1fbt52l13hl2_rtlv49ngykr0000gn/T/ipykernel_27745/1840109679.py:4: InterpolationWarning: The test st
atistic is outside of the range of p-values available in the
look-up table. The actual p-value is greater than the p-value returned.

  result = kpss(df)
```

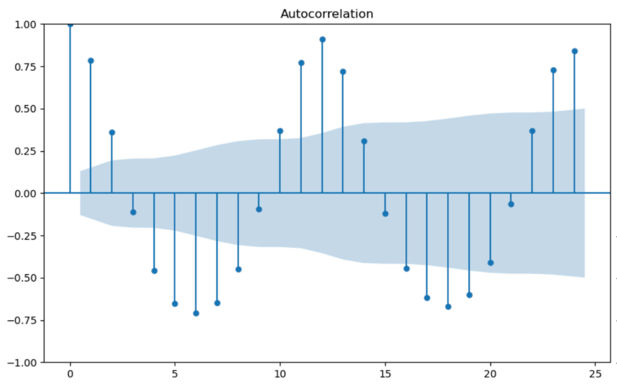
KPSS Test for Stationarity

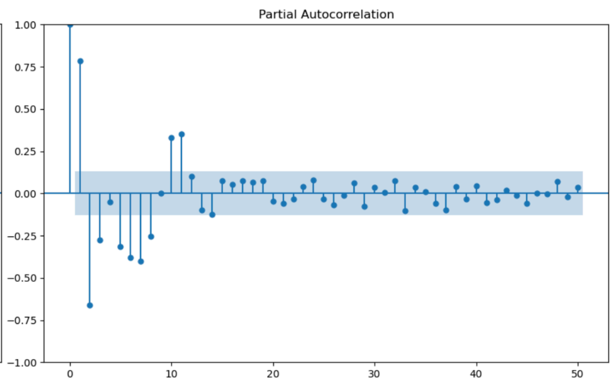


```
Results of Dickey Fuller Test:
Test Statistic                  -3.160544
p-value                          0.022385
#Lags Used                      13.000000
Number of Observations Used    215.000000
Critical Value (1%)             -3.461136
Critical Value (5%)             -2.875079
Critical Value (10%)            -2.573986
dtype: float64
```

ADF Test for Stationarity

We also obtain the following ACF and PACF plots:

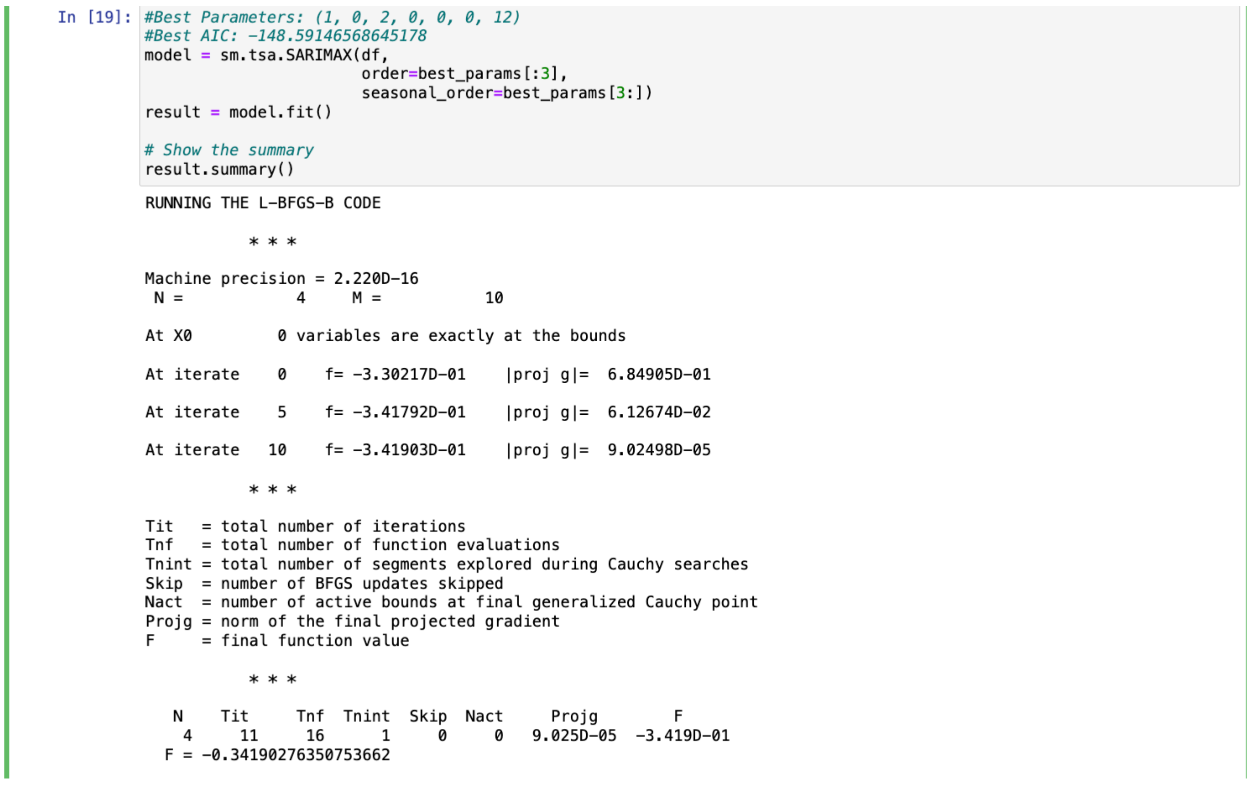




Auto-Correlation Factor Plot

Partial Auto-Correlation Factor Plot

Finally, we do a grid search for the parameters for the SARIMA model. One important thing to keep in mind here is we need 7 parameters instead of the original 3 needed in ARIMA. Of the 4 additional parameters, 3 are the seasonal counterparts of Auto-Regressive Order, Differencing Order and Moving Average Order. The final parameter is the seasonal period, which is 12 in our case as after every 12 months, the temperature cycle repeats itself. After the grid search is completed, we get the parameters and using these parameters, construct the SARIMA Model. Finally the model diagnostics were obtained and its performance proved to be satisfactory.

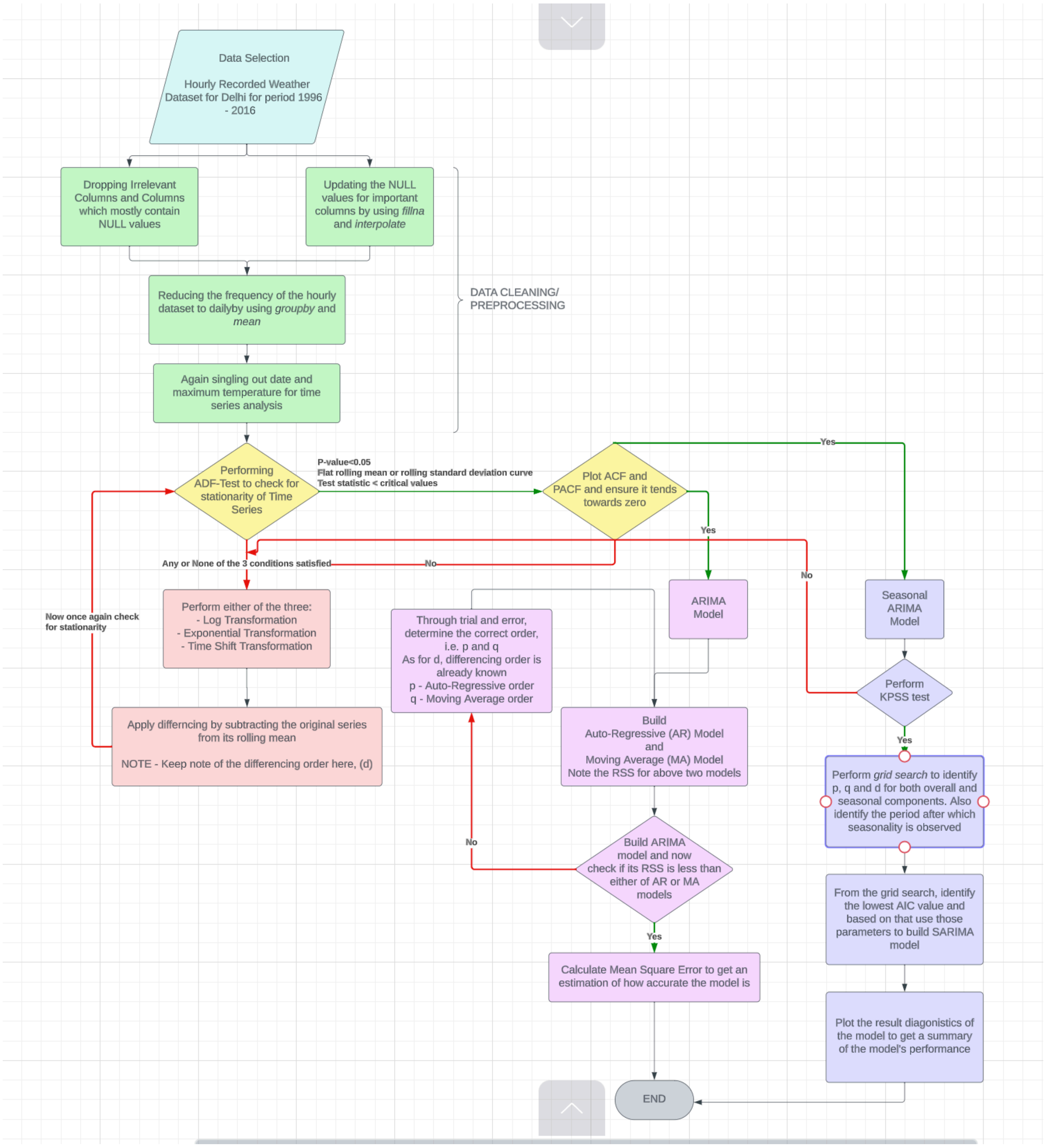


```
In [19]: #Best Parameters: (1, 0, 2, 0, 0, 0, 12)
         #Best AIC: -148.59146568645178
         model = sm.tsa.SARIMAX(df,
                                order=best_params[:3],
                                seasonal_order=best_params[3:])
         result = model.fit()

         # Show the summary
         result.summary()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            4     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f= -3.30217D-01    |proj g|=  6.84905D-01

At iterate    5    f= -3.41792D-01    |proj g|=  6.12674D-02

At iterate   10    f= -3.41903D-01    |proj g|=  9.02498D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    4     11     16      1     0     0   9.025D-05  -3.419D-01
  F = -0.34190276350753662
```

SARIMA Model Building Process

We stop at this point for now. For getting more accurate predictions and building better models, we need a bit more research to do in the field of ARIMA and SARIMA models. This is further discussed in the Future Work Section. Summarising the whole process, first of all we obtained the dataset from Kaggle for hourly weather of New Delhi from 1996-2017. We first of all removed the data of 2017 to keep it for testing purposes. Then we cleaned the dataset by removing columns with many Null values or useless columns. Then we transformed the frequency of the dataset to daily by averaging out the hourly metrics for each day, Then we only took out the date and maximum temperature columns as we only need those two for Time Series Analysis. Then we checked the stationarity of time series with multiple transformations and multiple degrees of differencing. Finally we built the ARIMA and SARIMA model and get their performance metrics. For now, the below flowchart gives a good idea of the process followed in this methodology.


Data Selection
Hourly Recorded Weather Dataset for Delhi for period 1996 - 2016
Dropping Irrelevant Columns and Columns which mostly contain NULL values
Updating the NULL values for important columns by using fillna and interpolate
Reducing the frequency of the hourly dataset to dailyby using groupby and mean
Again singling out date and maximum temperature for time series analysis
DATA CLEANING/ PREPROCESSING
Performing ADF-Test to check for stationarity of Time Series
P-value<0.05
Flat rolling mean or rolling standard deviation curve
Test statistic < critical values
Plot ACF and PACF and ensure it tends towards zero
Yes
Yes
No
Any or None of the 3 conditions satisfied
No
Now once again check for stationarity
Perform either of the three:
- Log Transformation
- Exponential Transformation
- Time Shift Transformation
Apply differncing by subtracting the original series from its rolling mean
NOTE - Keep note of the differencing order here, (d)
Through trial and error, determine the correct order, i.e. p and q
As for d, differencing order is already known
p - Auto-Regressive order
q - Moving Average order
ARIMA Model
Build Auto-Regressive (AR) Model and Moving Average (MA) Model
Note the RSS for above two models
Build ARIMA model and now check if its RSS is less than either of AR or MA models
No
Yes
Calculate Mean Square Error to get an estimation of how accurate the model is
END
Seasonal ARIMA Model
Perform KPSS test
Yes
Perform grid search to identify p, q and d for both overall and seasonal components. Also identify the period after which seasonality is observed
From the grid search, identify the lowest AIC value and based on that use those parameters to build SARIMA model
Plot the result diagionistics of the model to get a summary of the model's performance

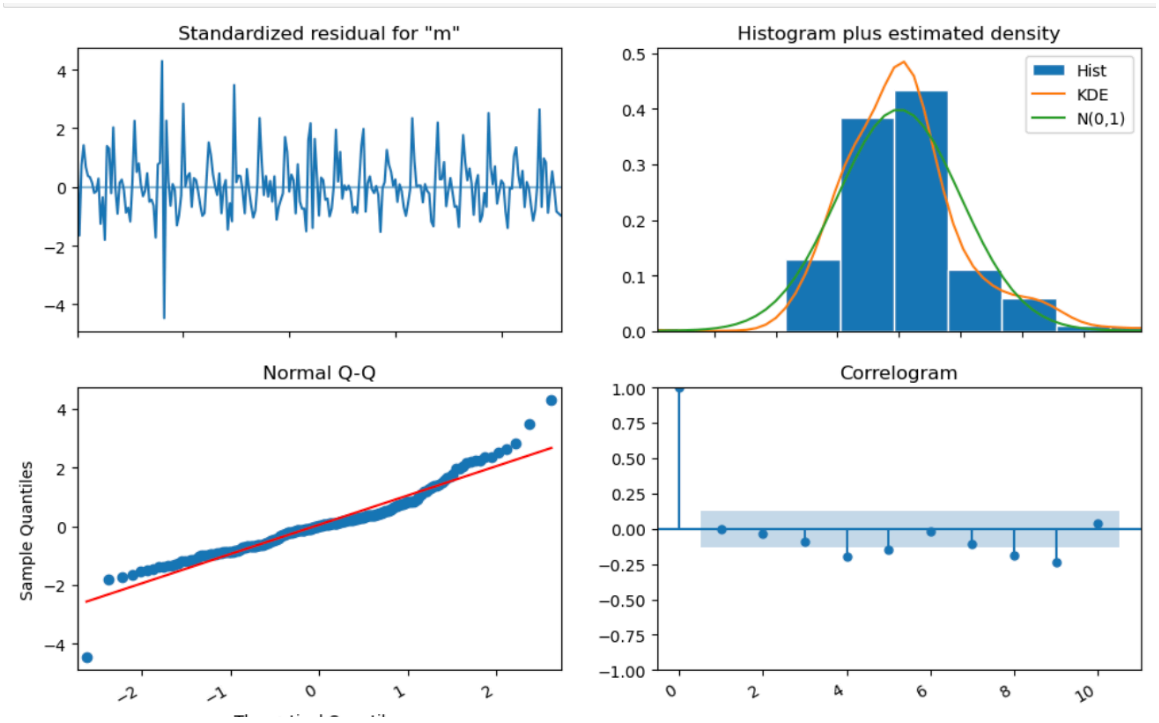
FlowChart for Proposed Methodology

RESULTS

So we built three different models, two ARIMA models and one Seasonal ARIMA model. The first ARIMA model was produced from a stationary time series which was first order differenced and then exponential decay transformation was applied on it. The complexity of this differencing and transformation made it difficult to reverse this transformation, and hence we had to apply this same transformation on the original time series to compare it with the predicted time series. After building the ARIMA model, we got a **Mean Square Root Error of 0.0126**. While this may seem quite low, it is imperative to keep in mind the dataset is already difference and exponential transformed, so a low value was quite within our expectations.

The second ARIMA model made use of a simpler time series. This time series had no differencing, i.e. order of differencing was 0 here. Only a single log transformation was applied to at an attempt to flatten the curves and get a better shot at stationarity. While the rolling mean was still no close to flat, the ADF tests were about satisfactory, and we went ahead and built an ARIMA model. For this ARIMA model, the RSS value was lesser than the one with the exponential decay model, suggesting this might perform better. Then we reverse-transformed the predicted time-series using exponential transformation and compared it with the original dataset, producing a **Mean Square Root Error of 2.9**. In other words, the actual and predicted daily maximum temperature had a margin of about 3 degrees.

Finally, we shifted to a Seasonal ARIMA Model. Due to the ability of this model to study trends, repeating patterns and seasonality better, this should perform better as the temperature cycle repeats after 365 days/12 months. We also used a time series with different frequency, monthly instead of daily for this one. After performing a grid search for the 7 parameters and building the SARIMA model, the diagnostics report produced the following results.



SARIMA Model Diagnostics

Comparing our results to existing work, this report unfortunately cannot claim any sort of upper hand over existing work. Superior models with a lot of intensive math behind stationarity and the model parameter calculations have been employed to minimise the MSE to below 1. For example, the National Weather Services (NWS) in the United States makes use of sophisticated SARIMA models to generate short term weather forecast. These models have also been used for predicting wind speed, solar radiation and humidity. In [11], the monthly mean temperature of 1951-2014 was studied in China using machine learning, and then a SARIMA model was built and predictions were made for 2015-2017. It produced an MSE of 0.89, suggesting very good accuracy.

CONCLUSION

After studying a lot of research papers on the use of Machine Learning in the field of weather forecasting, and trying out my own models, I can definitely say that it is a promising aspect in this field. The field of weather forecasting has been dominated by Numerical Weather Prediction (NWP) Models, and rightly so because these models, while they are complicated, they account for multiple environmental factors to produce accurate results. These accurate results are the reason they are so reliable. However, as discussed in the last part of the Results section, short term weather prediction has found ARIMA and SARIMA model to produce promising results, so much to the extent that they have even found their use. From short-term temperature and precipitation prediction to predicting wind speed for renewable energy and predicting solar radiation and humidity for agricultural purposes, these machine learning techniques are making strides in this field.

The simplicity and computational efficiency provided by these models over NWP make it less intensive and produce results in less time. Also they do not require much data and integrating these models with Machine Leaning Pre-processing techniques gives us much better chances at handling missing values. Machine Learning techniques can also be employed to study hidden patterns and trends in the time series better. While this report couldn't produce much promising results, due to time constraints, the field definitely provides a new avenue for weather forecasting. There have been multiple research papers such as [11] which have produced impressive results.

However, there lies a big drawback with machine learning techniques and that is their inability to include the various factors affecting weather. Part of the reason why NWP models are preferred because they factor in the Earth's climate cycle, greenhouse emissions and most importantly the side-effects of human activities on climate such as global warming. Take the case of Delhi, for example, we can't expect ARIMA or SARIMA models to predict weather during winter properly because it would be quite difficult for it to account for the all the pollution involved during

Diwali weather, and the crop burning and smog and all such stuff, all of which are a direct consequence of human activities.

FUTURE WORK

There is a lot of scope for improvement in this piece of report. The only actual landmark I believe this report achieved is the getting a stationary time-series. Lots of transformations and differencing was applied to get to that point. However for the steps succeeding that, a lot of more research and work needs to be done. Starting from the parameter selection itself, it would be quite more better to understand the math behind it, so that we can infer the suitable parameters to apply from our dataset.

And then we also need to study the different performance metrics for these models. Relying on a few parameters such as RSS and MSE only is unreliable and doesn't help us evaluate the model from all perspectives. Then the final part is getting the predictions. While drawing out the predictions for the ARIMA and SARIMA models, it was clearly evident that either the models struggled to identify the seasonality or trends within the data, or they couldn't translate the recorded features of the time series to future predictions. This section, in particular is one which requires a lot of work.

After all the above tasks are done, and we are able to draw out more reasonable predictions, close to actual temperatures, then only we can start working on introducing other parameters that would hint at human-induced effects on weather, such as Air Quality Index. In summary, a lot of work needs to be done, but once we can produce ARIMA/SARIMA models which produce results with good accuracy, then we can start looking for ways to make these models not only study environmental and climate data for the past years, but also recently introduced parameters that give us a better insight into the human activities that shape the climate. One way would be to use neural networks to build a relation between different human activities and weather features.

REFERENCES

[1] – Deepti Mishra and Pratibha Joshi: A Comprehensive Study on Weather Forecasting using Machine Learning, in 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)
Amity University, Noida, India. Sep 3-4, 2021, Source: IEEE Xplore

[2] – Elias Dristas, Maria Trigka and Phivos Mylonas: A Multi-class Classification Approach for Weather Forecasting with Machine Learning Techniques, in 17th International Workshop on Semantic and Social Media Adaptation & Personalization (SMAP), 2022, Source: IEEE Xplore

[3] – Swati Nigam, Monica Gupta, Akshaya Shirnivasan, Araveti Venkatej Sai uttej, Chaitra Kumari and Disha P B: Comparative Study to determine Accuracy for Weather Prediction using Machine Learning, in 2023 International Conference on Computer Communication and Informatics (ICCCI), Jan 23-25, 2023, Coimbatore, India, Source: IEEE Xplore

[4] – R.Sathya, Arpit Rastogi, Ankit Kumar and Shubham Singh: Weather Based Future Rain Prediction using Machine Learning with Flask Framework, in 2022 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), Source: IEEE Xplore

[5] – Mark Holmstrom, Dylan Liu and Christopher Vo: Machine Learning Applied to Weather Forecasting , Source: Stanford University

[6] – Risul Islam Rasel, Nasrin Sultana and Phayung Meesad: An Application of Data Mining and Machine Learning for Weather Forecasting, Source: ResearchGate

[7] – Shruti Dadhik, Vibhakar Pathak, Rohit Mittal and Rohit Doshi: Machine Learning for Weather Forecasting, Source: Machine Learning for Sustainable Development

[8] – Mihir Bhawsar, Vandan Tewari and Preeti Khare: A Survey of Weather Forecasting based on Machine Learning and Deep Learning Techniques, ISSN 2347 – 3983, Source: http://www.warse.org/IJETER/static/pdf/file/ijeter24972021.pdf

[9] – Ashfaq Ali Shafin: Machine Learning Approach to forecast Average Weather Temperature of Bangladesh, Online ISSN: 0975-4172 & Print ISSN: 0975-4350, Source: Global Journals

[10] – Mehmet Tektas: Weather Forecasting using ANFIS and ARIMA models. A Case Study for Istanbul, Source: ResearchGate

[11] - Ping Chen, Aichen Niu, Duanyang Liu, Wei Jiang, Bin Ma: Time Series Forecasting of Temperatures using SARIMA: An Example from Nanjing, IOP Conference. Series Materials Science and Engineering, August 2018, Source: ResearchGate