# PREDICTING WEATHER INFORMATION USING MACHINE LEARNING METHODS AND ARTIFICIAL INTELLIGENCE

RAVNEET SINGH, MSC MACHINE LEARNING

FINAL-THESIS REPORT

MAY 2021

**TABLE OF CONTENTS**

**DECLARATION**

I declare that this thesis work has been submitted by me only at Liverpool John Moores University as part of Master's degree in Machine Learning and Artificial Intellgience. Current thesis work as from my knowledge has not been presented or accepted in any other institutional body till now. I also declare that this thesis work is done through my own investigation and study. Appropriate references used in the study have been cited including the data set used for implementation.

Ravneet Singh
India, 2021

**ACKNOWLEDGMENT**

I feel myself privileged to be part of Master's program in Machine Learning and Artificial intellgence from Liverpool John Moores University. It has been a great learning experience. I would further like to acknowledge the role of my Thesis supervisor Korupalli V Rajesh Kumar in delievery of this Theisis. His indepth review of the Thesis work and guidence has been a crutial factor in delivering this Thesis in good quality. I would also like to thank the freely available Machine learning material and data sets without which this work would have been very difficult to accomplish.

Ravneet Singh
India, 2021

**ABSTRACT**

Weather forecast finds its application in wide variety of fields, industries and sectors. Its not only limited to these areas but find its use and application in day to day activities. It can be used for planning human activities and efforts better. Within Weather forecast we would be focusing on the temperature forecasting. Environment temperature is something which effects all the human being living on this planet. Abrupt or unexpected change in temperature is something which can have large scale effect on living beings as well as linked sectors and crucial organizations. Agriculture sector is one of the many which we can name. Temperature change can also signify natural calamities or any disasters which can be on its way. There are many organizations which studies the changes in temperature which are happening in specific areas. Often its too late to do any preventive steps once such changes are encountered. Forecast of temperature changes can thus help to foresee such condition and allows respective organization to take preventive steps on time. Traditionally there were many techniques which were used to forecast temperature. However, there were compromises on accuracy of the results. With growth of Artificial Intelligence and Machine learning algorithms there have been a new surge in area of forecasting. Machine learning algorithms if properly trained gives very good accuracy in comparison to the traditional prediction methods. In this study we will go through the weather forecasting history, current work which is done in the area of temperature forecasting. We will study and implement temperature forecasting on Delhi, India data set using time series prediction. Our aim is to do proper data cleaning and preprocessing and then apply Neural Network and ARIMA (autoregressive integrated moving average model) on this data set. We look at the complexity of the modelling process, parameter tuning and resource consumptions. Evaluation metrics for time series data will be used to analyze the results and accuracy. Accuracy and forecast power of both models are compared and results are presented.

**LIST OF TABLES**

**LIST OF FIGURES**

# CHAPTER 1: INTRODUCTION

## 1.1 Background of the Study

With increase in urbanization and other human intervention in nature, there have been vast change in weather and environmental conditions. Below paper provides a good summary of how urbanization is affecting the temperature conditions (Sarah Chapman, 2017). Predicting of weather condition has applicability in wide variety of fields. Various natural phenomenon like cyclones, rainfall, draughts, sea conditions etc. has wide range of effects on daily life. Predicting the weather condition with high precision is important to avoid huge loss to the various dependent fields like crops and agriculture, irrigation department, forest department, travel domain, environmental science etc. In our day-to-day activity having a prediction of weather condition is highly helpful to plan our tasks with efficiency and better output. It can also help to take care life of the living beings on this planet better and can convey information to others who are in danger due to extreme weather conditions. Due to recent changes in climate/weather condition this prediction gives us pointers for taking preventive measure to restore our climate and slow its deterioration. It is very important to train the ML algorithm with high quality data set which has been thoroughly cleaned and preprocessed. This should ensure that quality of prediction is good on unseen data. Weather data can be retrieved from vast variety of sources such has climate departments, satellite data sets etc.

Agriculture is one of the main departments which heavily depends on climate/weather prediction. Appropriate steps can be taken for marketing, storage and harvesting of the crops based upon these predications. In Paper (S.Veenadhari, 2014) a software tool named ‘Crop Advisor’ is created to indicate influence of weather conditions on Crop yield. Here various climate/weather parameters are considered and individual effect on the Crop yield is predicted. Predication accuracy of developed model on soyabean crop has been provided. Another impacting area is the cloud burst, which can lead to excessive amount of rainfall and can have destructive effect on various industries. Paper (Pabreja, 2012) develops models for prediction of cloudbursts using clustering techniques. It works on the moisture distribution, vertical motion, sheer and creates a model which can do prediction of 16 days in advance. Having information of Storms in particular area is also one of the major applications of weather prediction. Accurate prediction of stormy condition prior in time can help us take preventive steps which proves to be very helpful to save lives and related damages. Wind related parameters proves to be really helpful in such predictions. Live

sensors are often used to measure these parameters on real time which are used as input to the trained model to get the forecasting results.

Convolution Neural Network is also often used as a feature extractor, output of which is fed into a Recurring neural network. Paper (Jennifer L. Cardona, 2019) explores the video capture to predict the same. This paper also brings forward the usage of LSTM (long short-term memory) networks for prediction of turbulent wind speeds. Paper (Jonathan A. Weyn, 2020)works on weather prediction using Deep Convolution Neural Network on a cubed sphere to predict various atmospheric variables. Though this model is less accurate than other forecasting models, but it provides a good comparative study of implementation on given data set. There are many studies which are done in weather prediction on the previous years-maintained data. Along with various data mining and other traditional techniques, with growth of Machine learning and Artificial Intelligence there has been considerable increase in prediction accuracy and methods used for this purpose. Below paper explores data mining techniques for weather prediction (P.Kalaiyarasi, 2018). It researches on available methods for weather prediction and goes through the data mining stages. There are researches which work on predicting weather condition and predicting rainfall, this research uses real time data through sensors and use the data set to get the prediction on rainfall levels (Gopinath N, 2020). There are also researches like (ZhanJie Wang, 2017) on Weather forecast and Cloud computing. Cloud computing adds the advantages of using the efficient, secure, reliable data storage and therefore reduces cost and infrastructure required for weather data storage. Paper (Garima Jain, 2016) explores the time series and correlation models to understand the weather forecast. Time series analysis help to understand and build the stochastic nature of the model from given observational series. Further it helps us to forecast or predict values on the unseen data from the previous data observations. Decision trees have been used to predict the weather forecast, a research (Kumar, 2013) explores the usage of decision trees and brings forward various techniques like Gini Index and information gain for the prediction of weather information. Research is also being made on Regressive models to predict the weather conditions, paper (N Anusha, 2019) executes a Regression model to predict the rainfall conditions. With advance of the Machine learning algorithms there have been research on the Artificial Neural networks, below paper bring forward this approach (Priyanka Mahajan, 2017). Approach to handle weather prediction using Artificial Neural Network is discussed. Since weather data is generally nonlinear and require a complex model to formulate the prediction on the data set, Neural Network is often a good choice.

## 1.2 Problem Statement

We are using machine learning algorithms and artificial intelligence to solve the problem of predicting weather information that is temperature on a data set from Delhi, India.
To forecast weather conditions, machine learning algorithms Artificial Neural Networks and ARIMA (Autoregressive integration moving average) will be used.
We will explain why, and which model provides good predictions in terms of time and accuracy based on the evaluation metrics. We will also note fine-tuned hyper parameters and model structure, all of which can aid us in delivering accurate results on unknown data.

## 1.3 Aim and Objective

The main goal of this study is to propose a machine learning model for weather prediction using Neural Networks and ARIMA (autoregressive integrated moving average) using data from Delhi, India. Determine the weather situation on the test data set and compare the accuracy of the predictions.
The model's performance will be assessed after it has been trained on a clean and pre-processed data collection.
The research objectives are focused on the study's intent, which is as follows:

- To examine the trend and relationship between different weather variables such as temperature, humidity, and wind speed, among others.
- Check for data imbalances and missing values and use appropriate preprocessing techniques to deal with them if they exist.
- Compare the predictive models' output metrics and determine which model is better suited for the weather data.

## 1.4 Scope of the Study

The study's scope will include:
Analyzing the data set and performing any necessary cleaning and preprocessing.
On the given data set, build Artificial Neural Network models and Autoregressive integrated moving average model.
Perform hyper parameter tuning to achieve the highest level of precision possible with the available resources.

Compare and analyze the models used, emphasizing the accuracy achieved on the given model structure.

## 1.5 Significance of the Study

To predict the weather in Delhi, India, this study will use machine learning models such as Neural Networks and Autoregressive integrated moving average.
The resulting model can be used on unknown test data to determine weather conditions, and the output can be used to make important planning decisions.
We will try to fine-tune our model using the predictive model's various hyperparameters.
The model with the better accuracy for the weather forecast data collection will be determined by the accuracy comparison.
It will also assist us in comparing the structure of the model, such as the number of layers and neurons used in Neural Networks, in order to strike a balance between the hyperparameters' complexity and the performance obtained.

## 1.6 Structure of the Study

Below is how we have structured the study. As we saw the first chapter is the one which provides the introduction to the report. It provides the background of the study, gives details of the problem statement which we are going to handle. It brings forward the aims and objectives of the study. We clearly explain what we want to achieve out of this study. Chapter 1 is aiming to provide the introduction and detail of the problem we are handling. It includes what is our aim and objective in taking this study. It ends with providing the use cases and significance this study holds for the end users and machine learning society.
Chapter 2 is focused on the literature review on the domain this study is based upon. It checks the previous year research papers and highlights the work which has been done in past around this area. It goes through different type of problem statements which are possible around this study. How currently these problems are handled. It also shares the results that are being achieved till now for the problems in this domain. It has section explaining in detail about the details of data mining in the weather prediction area. Then it narrow downs and bring forward various weather forecast which has been taken by machine learning algorithms. We will go through the Temperature prediction, Rainfall prediction and crops

prediction using machine learning. The achievement and results till current date has been analyzed and put forward in this section.

Chapter 3 focuses on the Research Methodology we would like to follow for our implementation of the study. It starts with the tools we would be using for the implementation work. We will further discuss the data set we have chosen for our study. Then we will go forward and discuss the preprocessing steps we would like to go over during the implementation phase of the study. In preprocessing we have gone through missing values treatment, checking of outliers, how do we plan to handle the categorical variables and scaling which we would like to do on the data set. We have then a section on Exploratory data analysis where we highlighted its importance on our data set understanding. Further we will discuss about the data sampling and how we would be implementing it on our data set. Since our study is focused on time series forecast next sub section we will highlight the points in time series data prediction. Further section we will go through the Evaluation metrics which are available to us and which we will follow during the implementation phase. Next is the modelling section where we will discuss about the two models which we will be implementing. This section we will go over the Neural networks and ARIMA modelling steps. We will highlight about the modelling procedures and steps we would be taking during the modelling including the parameter tuning which is essential for good results.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

Weather has been playing an important role in day-to-day life activities. The impact of weather is enormous on our basic needs. Various fields highly depend upon the weather conditions and requires this to work in appropriation. Given the gathered data with various weather parameters such as temperature, humidity, dew levels etc. machine learning algorithms can be used to determine or predict the weather condition. This prediction can be used in many industries to better frame and plan their execution and take required steps to deal with abnormal conditions.

We will see and understand ML models which can be used to do this prediction. We will go over the literature and past research papers in detail and understand the work that has been done and analyzed in this area. We will look through various sub level problems of weather prediction and see how current researches have approached these problems using Machine Learning algorithms. We will also see the impact of these researches on the end user and various sectors, organizations.

## 2.2 Data Analytics in weather prediction

We do see in day-to-day life prediction of weather information which has wide variety of uses. Current inventions and algorithms use data analytics to perform these weather prediction tasks. In general Data Analytics uses predictive modelling and machine learning are used to make a relation between available data and relative predictors. With increase in development of complex machine learning models the accuracy of the prediction using past data has significantly increased. To obtain the results it is often significant to have right data in quantity and quality. Weather data is considered and closely related to time and location it is taken on. Along with data received through various sensors, satellites with advancement of mobile device there has been a positive revolution in data collection and their utilization using machine learning and artificial intelligence. Analysis of weather data can help in various fields like, Agriculture: with due increase in population this field is one of the prime importance for the government across the world. Weather prediction can help to come up with necessary laws and schemes so as to help the farmers grow and maximize their yield. Having accurate prediction of rainfall, droughts can help to overcome major future problems.

This paper (Journal and Technology, 2020) highlights the predictive analysis using machine learning. This discusses about models like linear regression, Ordinary least squares, and importance of data preprocessing in prediction process.

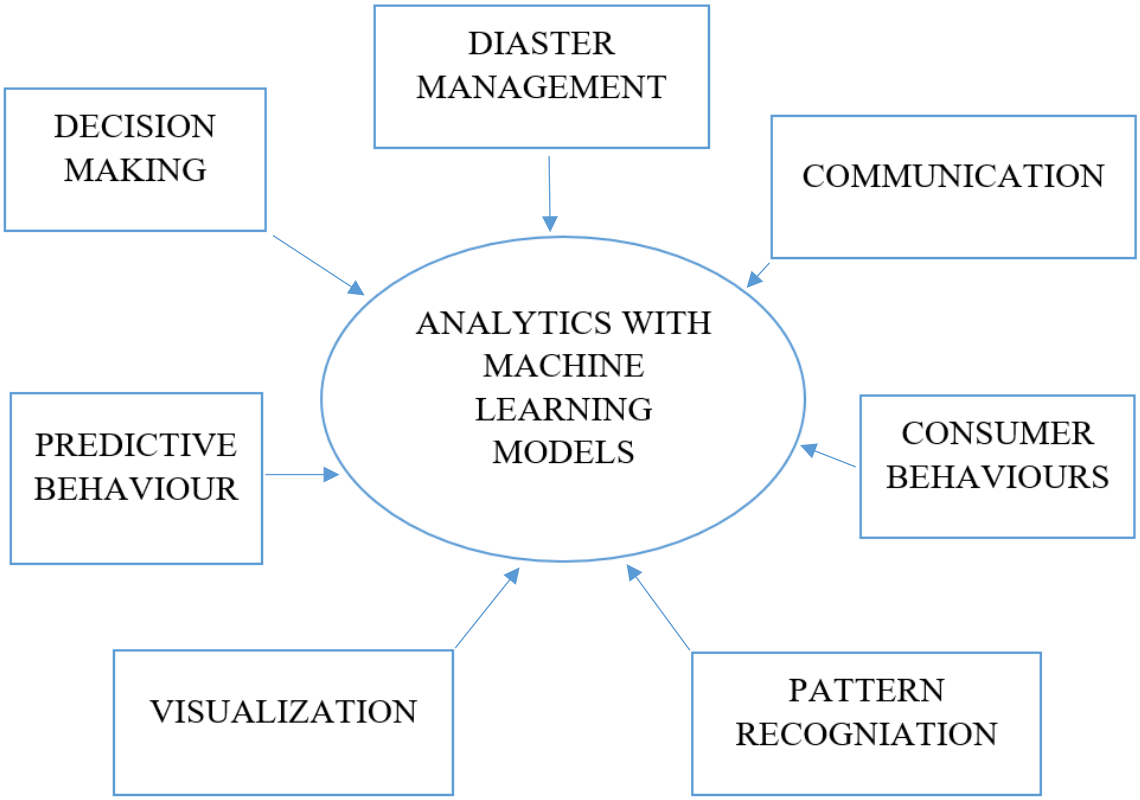



Figure 2.1 : Some of the use cases of Analytics using Machine learning modelling.

Paper (Shinde et al., 2017) discuss about predictive analysis using R Analytical tool. This is an open-source statistical tool which can help in predictive analysis. Many different fields including agriculture generates large amount of data which is often referred as Big Data. This data is very useful in predictive analysis on tools like this. Various sectors use such tools to take many decision tasks. This paper shows the implementing algorithms such as Random Forest, Latent Dirichlet Allocation. A case study is also done which shows higher accuracy for Random Forest.

## 2.3 Data Mining in weather prediction

Both Predictive and Descriptive data mining techniques can be used for statistical analysis and predictions using captured weather data. Analysis can help to gather interesting facts without having any previous hypothesis. Effects of the weather conditions like future temperature if predicted can help to derive many after math decisions. Following paper (Biradar et al., 2017) explores the Naive Byes for classification and K Medoids algorithm for clustering for weather forecasts. It shows that data mining algorithms gives good results in comparison to the traditional algorithms.


Figure 2.2 : Applying data mining on climate data.

This (Dhore et al., 2017) research paper builds a numerical prediction model. It takes input climate data information which has history of complex climate parameters like precipitation, temperature, humidity etc. It generates patterns while training and then uses these patterns for the prediction process. FP growth algorithm is used in this paper. The output gives the predicted weather with temperature and effects of temperature can also be analyzed. Below are the few techniques which are most frequently used:

- Genetic Algorithms
- Neural Networks
- Clustering method analysis
- Memory based methods
- Regression
- Tree based methods.
- Bagging
- Rule Induction

Figure 2.3 : Data Mining Techniques.

Research paper (Muqeem and Javed, 2016) shows a comparative approach for data mining techniques and provides a fairly good comparison listed below:

| **Parameter** | **ANN** | **DTN** | **Clustering** | **GA** |
|---|---|---|---|---|
| Perfor-mance | Maxi mum | High | Moderate | High |
| Supervised Learning | Yes | Yes | No | Yes |
| Unsupervi-sed Learning | Yes | No | Yes | Yes |
| Computatio n Speed | Fast | Fastest | Slow | Fast |
| Cost effective-ness | Yes | Yes | Yes | Yes |
| Accuracy | More | Less | More | Modera te |
| Complexity level | Comp lex | Less | Moderate | Compl ex |
| Fault Tolerance | Yes | Yes | Yes | Yes |
| Nature | Analy tical | Analyt ical | Descriptive | Analyti cal |
| Domain Area | Comp lex | Simple | Complex | Compl ex |
| Parallel Computing | Yes | Yes | No | Yes |

Table 2.1 : A comparison of data mining models.

The chosen data mining techniques should be able to

1.) Manage real time data.
2.) Coordinate data from various sources if required.
3.) Should be able to handle data from various sources (sensors, pdf, csv etc)
4.) Should recognize the data redundancy.
5.) Interactive using Graphical interface.
6.) Should be able to handle visual data and also provide visual analysis of data.

7.) Cost effective, time efficient.

Below paper does a review of data mining techniques and application. (Liao et al., 2012) . paper analysis that between period of 2000 to 2011 around 14 thousand articles were found which uses above data mining techniques. Trend and usage of data mining techniques increasing ever since.

## 2.4 Weather forecast – Temperature prediction

Temperature predication has applicability in large variety of fields like agriculture, crop yields etc. Accurate temperature prediction is hence very useful in making some critical decisions based upon the field it is used in. Temperature can help to check the global warming state and also the effect of current lifestyle is having on world temperature and climate. Temperature data is collected from various sources and with advancement of wireless devices, they are often used to monitor the parameters in contrast to the wired system. Monitoring is often carried out in remote areas where through wireless communication the sensor data is transmitted to the base monitoring network. Below paper (Sutar, 2020) highlights Zigbee communication which is a standard for wireless communication, which uses above approach. Often a Graphical user interface is deployed at the base monitoring station to note the parameters sensed. Paper (Anjali et al., 2019) uses the data collected from Kerala, India during the period of 2007 to 2015 to predict the temperature. It uses techniques like SVM, Artificial Neural Networks for the prediction. Parameters or features used are Humidity, wind speed, rainfall, dew point. We see here that Neural network tends to perform better than the SVM on the given data set.

Below paper (Cifuentes et al., 2020) takes on the air temperature prediction taking in account global and regional point of views. Another common application of machine learning in weather prediction is the hourly forecast of the temperature. Paper (Souza, 2000) forecast the hourly temperature on the basis of previously noted temperature and maximum and minimum temperature supplied by the weather services. On the basis of temperature forecast given at some hours, forecast can be done at these hours, while for remaining hours some kind of interpolation can be used. For hourly temperature prediction many approaches are possible some approach uses Neural networks where each output neuron predicts hour of a day. We can have total of 24 neurons in this case. Since we would need big neural network for this the model is often unstable. Another approach is to use only one output neuron to

forecast next hour value. Forecasting here should be sequentially that means one hour is forecasted than this is input again to the network to get the next hour of forecast. Often it was noticed that models were failing to understand the mathematical relationship amongst the previous data. Paper (Kapoor and Bedi, 2013) presents a sliding window algorithm which tries to model the dependency between the data and then tries to do prediction on basis of this. Here a best match window is created, and this is used for prediction. Window is used as weather condition may show dependency on the a given period instead of one observation. This was then tested on champawat city data, to predict the minimum, maximum temperature. The algorithm shows good accuracy apart for the months where conditions are highly unpredictable.

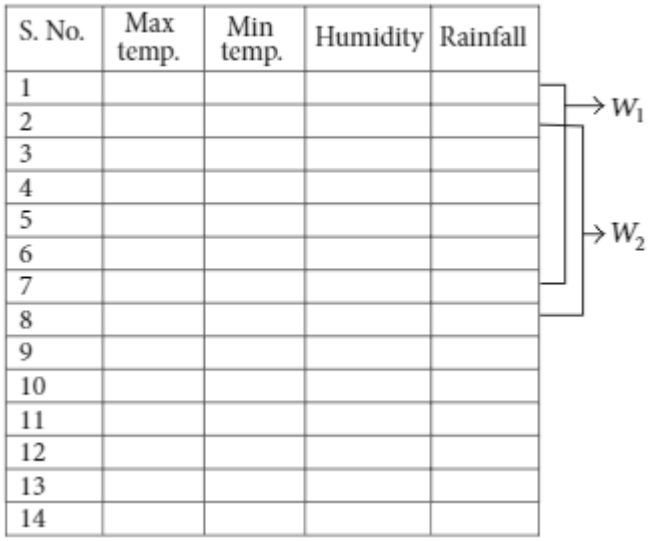



Figure 2 4 : Sliding window concept (W1, W2 represents window 1 and window 2)

Paper (Ganju, 2012) explores the prediction of temperature particularly to check for any avalanche forecasting. Artificial neural network is used for predicting maximum and minimum temperature on past twenty five years of maintained data set. Study suggests that artificial neural network are effective means to predict the maximum and minimum temperature for given data set. This paper (Corresponding, 2009) works on the Meteorological parameter forecast the data set has nonlinear structure and is known to be complex. An artificial neural network is designed to predict the summer monsoon temperature (minimum and maximum) in India. Results have shown that the prediction is good with less error.

## 2.5 Weather forecast – Rainfall prediction

Rainfall prediction is of very importance, because high rainfall can be very destructive on property as well as on the agriculture industry. Having an accurate rainfall prediction can help to take mitigation steps to deal with extreme conditions effectively. There are traditional hardware devices which can predict the rainfall however they are not much accurate. With advancement of machine learning accuracy has increased but we should use correct algorithm and proper data set. Paper (Mohammed et al., 2020) works on the regression for predicting the rainfall, data set used contains measurement of rainfall from 1902 to 2105 which include parameters such as amount of rainfall in mm, month etc. Since data set is large feature reduction is being used (PCA) here. Support vector machine and Multiple linear regression are used, it is seen that support vector machine gives more accurate results.

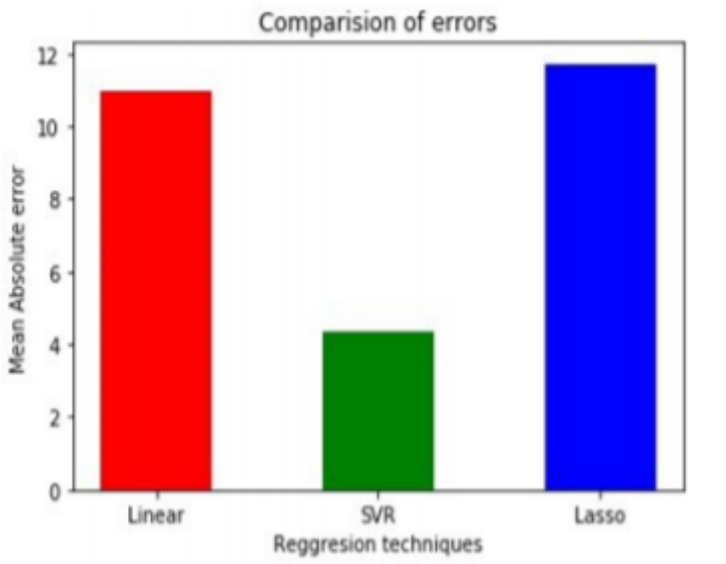



Figure 2.5 : Comparison among applied models.

Paper (Ridwan et al., 2020) list the case study on Terengganu in Malaysia. This uses ML algorithms like Bayesian Linear Regression, Boosting decision tree regression, Neural Network Regression. Two approaches were taken one, Rainfall prediction using Autocorrelation function second, Rainfall prediction using Projected error. First approach was shown to provide higher accuracy using Boosting decision tree regression. Below (Shah et al., 2018) paper studies the precipitation level in India. Paper also performs clustering technique with rainy days encountered and rainfall level as input. It predicts the rainfall level

consider other independent variables like temperature, humidity, wind speed etc. This paper shows that random forest is outperforming other modelling methods in terms of error rate.

## 2.6 Weather forecast – Crops

Agriculture is one of the main occupations specially in the rural areas. Government often needs to take best step in order to make sure Agriculture in the country is flourishing. It is very important that yield of the crops is up to the mark and as expected. Paper (Bangaru Kamatchi and Parvathi, 2020) describes the use of Machine learning in crop management. Earlier we note that traditional techniques were used to monitor the crop yield and growth. However, these are often restricted to certain crops or certain stage of cultivation and are often quite expensive on the other hand crop growth model provide more accurate results. Paper (Periyasamy and Devi, 2017) works on data set of Godavari district of Andra Pradesh India with records from 1955 to 2009. Data set include parameters like sowing in hectors, rainfall level and production in unit of tons. It uses the multiple linear regression to predict the crop yield along with density based clustering comparison. Papers presents and find density based clustering comparison to be suitable for these kind of data sets. Another paper (Majumdar et al., 2017) on clustering considers wheat crops and does clustering based upon temperature rainfall and soil type. Based upon these optimal parameters are derived for maximum crop yield. This paper uses random forest to predict crop yield, it works on Maharashtra India data which has five features precipitation, temperature, day type, vapor measurement, cloud detail. It also provides a good user-friendly Graphical user interface detailed below in figure 2 6:

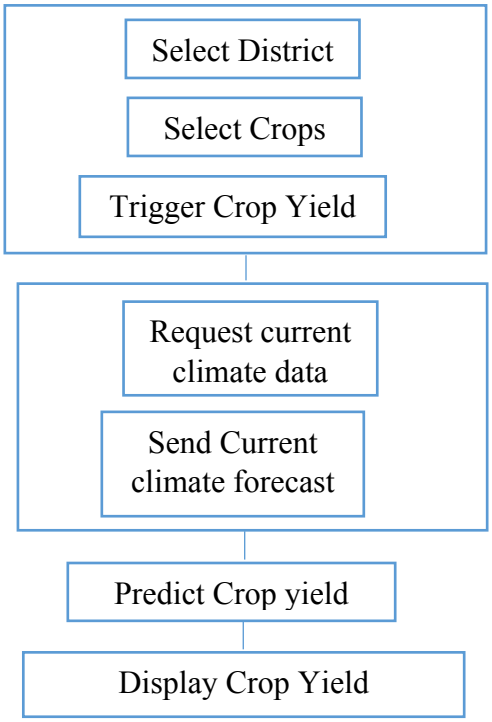



Figure 2.6 : Prediction for Crop Yield.

## 2.7 Summary

In this section we went through the literature review of weather forecasting. We saw and analyzed the past works which have been done in this Area. We presented that with the help of Artificial Intelligence and Machine Learning there have been made good progress in the area of weather forecasting. We saw how data analytics has given a whole new dimension to the area of weather forecasting. We briefly touch based upon the use cases of weather prediction and how Machine learning and data mining techniques has helped to cater these. We have gone through the Temperature forecasting and saw the difference of the traditional methods and the ML algorithms which helps in forecasting temperature. We saw some good comparison made by existing researches on different models. We also went through other areas like rainfall and crop forecasting. We saw few techniques and their impact on the current sectors mainly agriculture and meteorology. In the next section we would go through the detail of Research Methodology we will be using in our implementation of time series prediction of temperature on Delhi, India data set.

# CHAPTER 3: RESEARCH METHODOLOGY

## 3.1 Introduction

We will use a data collection from Delhi, India, and will apply machine learning model to forecast weather conditions. The primary focus will be on thoroughly inspecting and analyzing of collected data set, as well as cleaning and data pre-processing on the columns as needed. To perform prediction on the test data, modelling will be performed on given data set. The model structure and accuracy information will assist in evaluating the current model's behavior in forecasting weather data. This is a time series prediction for temperature feature. This paper (Tadayon and Iwashita, n.d.) suggests that Neural network has been known to work good for time series data especially LSTM which are known to remember information suits well for time series prediction in comparison to the classical models. We will analyze the data set to apply Neural Networks (LSTM, CNN-1D) and Autoregressive moving average model for prediction of temperature. We will further explore the building blocks of model architecture and steps to have good performance on each model. We will discuss proper evaluation metrics to compare the result and complexity of each model generated.

## 3.2 Research Methodology

### 3.2.1 Tools and Requirements

We will use python for implementation of the machine learning algorithms which will be discussed later. Windows machine on which anaconda software can be installed is used to write python code. Specifically, jupyter notebook will be used for coding and capturing relevant outputs from code execution.
Below are the libraries we will be using for implementation of our work:

| **Base Library/API Name** | **Usage Brief** |
|---|---|
| Numpy | For linear algebra and mathematics operation |
| Pandas | Reading and doing data analysis |

| | |
|---|---|
| Matplotlib, seaborn | For creating graphs and plots |
| Statsmodel | For operations required for ARIMA Modelling |
| Keras | For Neural Network Modelling |
| Sklearn | For data preprocessing, scaling, evaluation metrics computation |

Table 3.1 : Python base Library/API

We will be using the GPU access for training and doing hyper parameter tuning on the Neural Network implementation.

### 3.2.2 Data Selection

We have selected the Delhi data set which contains records maintained from year 1996 to 2017. Data is freely available from Kaggle at below location:
https://www.kaggle.com/mahirkukreja/delhi-weather-data
Below are the columns of the data set with description. We have total 20 columns present.

| Column Name | Column Description |
|---|---|
| _tempm | Temprature value in Celsius (°C) scale |
| datetime_utc | Data and time when record was noted |
| _conds | Weather condition like clear, cloudy etc |
| _dewptm | Dewpoint value |
| _fog | Indicates if weather was foggy or not |
| _hail | Indicates if hail was present or not |
| _heatindexm | Heat Index in Celsius (°C) scale |
| _hum | Humidity measurement |
| _precipm | Precipitation in mm scale |
| _pressurem | Pressure measured in milli bar |
| _rain | Indicates if rain was present or not |
| _snow | Indicates if snow was present or not |
| _thunder | Indicates if thunder was present or not |
| _tornado | Indicates if tornado was present or not |
| _vism | Visibility level |
| _wdird | Direction of the wind |
| _wdire | Wind direction in cardinal points |
| _wgustm | Measurement of sudden increase in wind speed |
| _windchillm | Measurement of wind chill factor |
| _wspdm | Measurement of wind speed |

Table 3.2 : Delhi weather data set details

### 3.2.3 Data Preprocessing

Data Preprocessing is one of the most important steps in machine learning model generation. If data is not well preprocessed then model might not show good accuracy levels. Based upon the data set we are using we will use below main steps of preprocessing.

#### 3.2.3.1 Missing values

Real time captured data often have lot of missing values, this can be data collection error or due to other factors like sensor might be malfunctioning and sometime miss to record the values. We loaded our data set using Pandas library in Python. Below is how the data set after reading looks like:

In [875]:
```
dataDf = pd.read_csv("testset.csv")
dataDf.head()
```

Out[875]:

| | datetime_utc | _conds | _dewptm | _fog | _hail | _heatindexm | _hum | _precipm | _pressurem | _rain | _snow | _tempm | _thunder | _tornado | _vism | _wdird |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 19961101-11:00 | Smoke | 9.0 | 0 | 0 | NaN | 27.0 | NaN | 1010.0 | 0 | 0 | 30.0 | 0 | 0 | 5.0 | 280.0 |
| 1 | 19961101-12:00 | Smoke | 10.0 | 0 | 0 | NaN | 32.0 | NaN | -9999.0 | 0 | 0 | 28.0 | 0 | 0 | NaN | 0.0 |
| 2 | 19961101-13:00 | Smoke | 11.0 | 0 | 0 | NaN | 44.0 | NaN | -9999.0 | 0 | 0 | 24.0 | 0 | 0 | NaN | 0.0 |
| 3 | 19961101-14:00 | Smoke | 10.0 | 0 | 0 | NaN | 41.0 | NaN | 1010.0 | 0 | 0 | 24.0 | 0 | 0 | 2.0 | 0.0 |
| 4 | 19961101-16:00 | Smoke | 11.0 | 0 | 0 | NaN | 47.0 | NaN | 1011.0 | 0 | 0 | 23.0 | 0 | 0 | 1.2 | 0.0 |

| _wdire | _wgustm | _windchillm | _wspdm |
|---|---|---|---|
| West | NaN | NaN | 7.4 |
| North | NaN | NaN | NaN |
| North | NaN | NaN | NaN |
| North | NaN | NaN | NaN |
| North | NaN | NaN | 0.0 |

Figure 3.1 : Delhi weather data top 5 records/rows.

From the first look of the data set we notice that there are some nan values which indicates that these values are missing in the data set.

Below are the appropriate steps which we can take to handle missing values:

1.) Analyse the mean, median, mode and standard deviation of a column. Check the data distribution and skewedness. Take a decision if we should impute the values with mean/median or mode (for categorical column). We don't want to bias our data set so before using this approach we would consider the number of values we are imputing. Therefore, will consider appropriate threshold for the percentage of missing values in a column when applying this method.

2.) We will analyse and remove columns which have more than 50 percentage of missing values.

3.) We will analyse and remove rows which have one or more nan values. As we will be losing data in this approach. We will make sure to apply this only in certain limit.

4.) Check if any of other methods needs to be used like k- nearest neighbour algorithm.

Below is how the percentage of missing values looks like on our data set. Please note that our data set has well over one lakh records/rows. we see we can apply above rules and clean

our data set to get rid of the nan/missing values in them. Based upon the missing value percentage we see that rules 1 to 3 should be enough.

```
In [876]: 1  round(100*(dataDf.isnull().sum()/len(dataDf.index)), 2)

Out[876]: datetime_utc      0.00
          _conds            0.07
          _dewptm           0.61
          _fog              0.00
          _hail             0.00
          _heatindexm      71.13
          _hum              0.75
          _precipm        100.00
          _pressurem        0.23
          _rain             0.00
          _snow             0.00
          _tempm            0.67
          _thunder          0.00
          _tornado          0.00
          _vism             4.38
          _wdird           14.61
          _wdire           14.61
          _wgustm          98.94
          _windchillm      99.43
          _wspdm            2.33
          dtype: float64
```

Figure 3.2 : Delhi weather  data set nan/null percentage per columns.

#### 3.2.3.2 Checking Outliers

Outliers are the values which are very different from the set of values for a column. Outliers can affect the modelling results significantly. During implementation we will check and take appropriate action for outlier treatment. Below are the steps which we can follow to analyze outliers:

1.) We will try to create visual box plots using python matplot library. Box plot are very useful to see the outliers for a given column.
2.) Further we will use inter quantile range to calculate the upper and lower bound. Values higher and lower than these bound will be removed.
3.) After removing these values, we will again create the box plot and see the reduction in outlier values.
4.) We will make sure that we don't remove too many records/rows while treating outliers. In case we find that there are too many outliers we will analyse the domain to make sure that these are really erroneous values.

5.) Further only if it is found that there are too many outliers which needs to be handled, we will try other techniques like imputation.

Below is the code in Python which provides visual view of number of outliers for the numeric columns. During the Implementation we will analyze further and use above points to take care of these outliers.

```
In [15]:
def checkOutliersBoxPlot(msg):
    plotNum = 1
    plt.figure(figsize=(35, 35))

    for col in dataDf.columns:
        if (dataDf[col].dtypes.name != 'object'):
            plt.subplot(5, 5, plotNum)
            sns.boxplot(dataDf[col])
            plt.title('Column {0} {1}\nTotal rows {2} '.format(col, msg, len(dataDf.axes[0])))
            plt.tight_layout()
            plotNum = plotNum + 1
    plt.show()
```

```
In [39]:
checkOutliersBoxPlot("Before Removing Outliers\n")
```

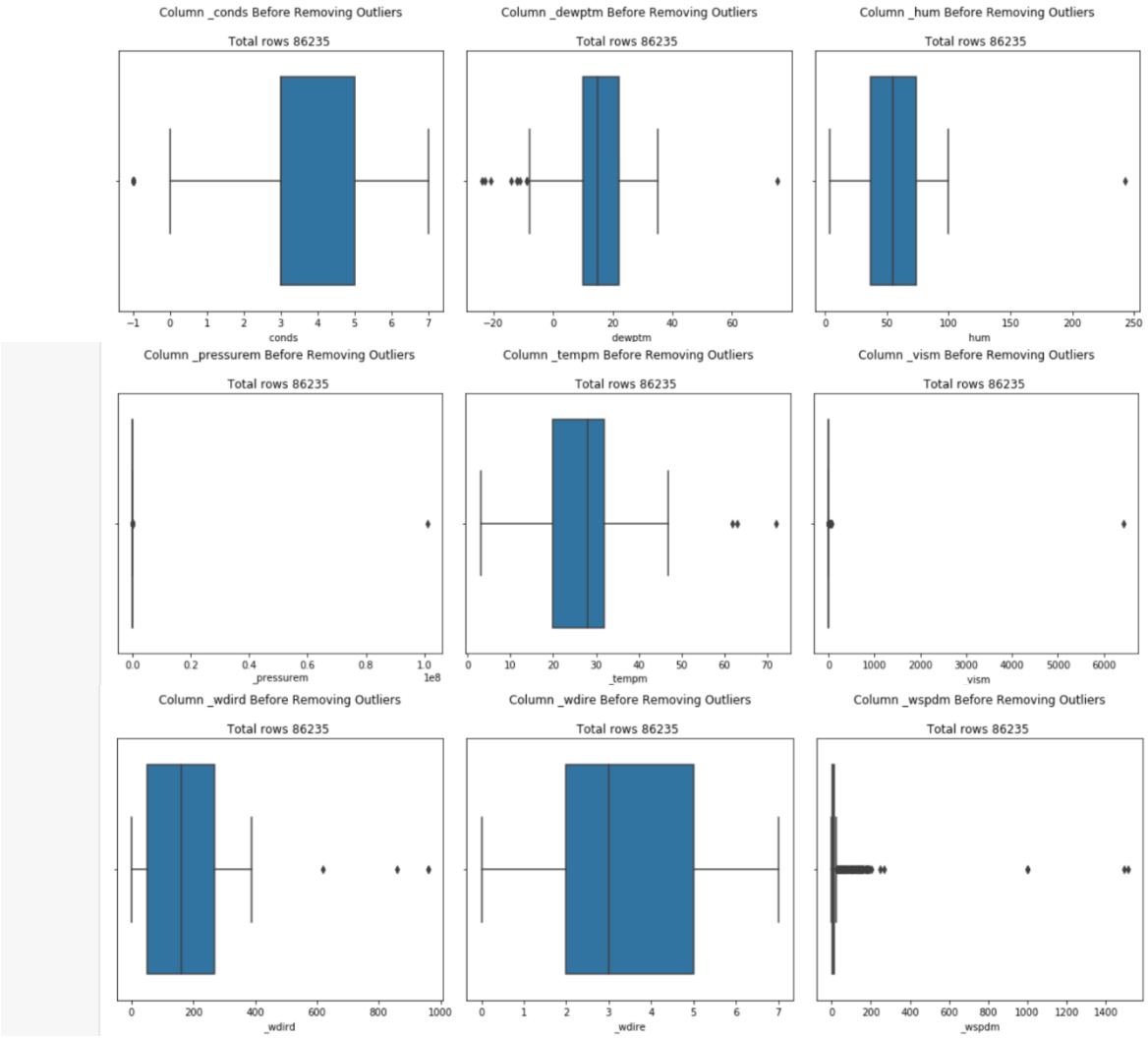



Figure 3.3 : Delhi weather data set box plot for checking outliers.

### 3.2.3.3 Categorical Variables

We will need to take steps to handle the categorical variables. As modelling requires data to be in numeric form. Checking our data set in Python we see that we have majorly three categorical columns (datetime_utc, _conds, _wdire). We will analyze the values of these columns and apply appropriate encoding to handle these categorical variables.

```
In [17]: 1 dataDf.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100990 entries, 0 to 100989
Data columns (total 20 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   datetime_utc  100990 non-null  object
 1    _conds       100918 non-null  object
 2    _dewptm      100369 non-null  float64
 3    _fog         100990 non-null  int64
 4    _hail        100990 non-null  int64
 5    _heatindexm  29155 non-null   float64
 6    _hum         100233 non-null  float64
 7    _precipm     0 non-null       float64
 8    _pressurem   100758 non-null  float64
 9    _rain        100990 non-null  int64
 10   _snow        100990 non-null  int64
 11   _tempm       100317 non-null  float64
 12   _thunder     100990 non-null  int64
 13   _tornado     100990 non-null  int64
 14   _vism        96562 non-null   float64
 15   _wdird       86235 non-null   float64
 16   _wdire       86235 non-null   object
 17   _wgustm      1072 non-null    float64
 18   _windchillm  579 non-null     float64
 19   _wspdm       98632 non-null   float64
dtypes: float64(11), int64(6), object(3)
memory usage: 15.4+ MB
```

Figure 3.4 : Delhi weather data set pandas data frame info() output.

Categorical variables can be handled in below ways :

1.) Using label encoding where we represent each value of categorical variable with numeric value. Generally, values are used in increasing order starting from zero.
2.) Use one hot encoding where each value of the categorical variable is converted into a column with value 0 or 1 indicating if the given row has this value or not.
3.) In case if categorical variable has values falling in bins like 0-20. We can provide a numeric value to each bin.

Below are the levels in the _conds categorical variable. We notice that there are few values which have very less occurrences. We will follow an approach to combine the values of the categorical variables so that occurrences are lowered. Example ‘shallow fog’, ‘heavy fog’,

‘patches of fog’, ‘partial fog’, ‘light fog’, ‘haze’, ‘light haze’, ‘mist’ can be combined into one ‘fog’. This will reduce the levels of the categorical variable. We will then use Label encoding to convert these to numeric values. Similarly, we will analyze the other categorical variable _wdire and follow above approach on this column.

```
In [21]: 1 dataDf['_conds'].astype('category').value_counts()

Out[21]: Haze                            47602
         Smoke                           20760
         Mist                             9375
         Clear                            3129
         Widespread Dust                  2856
         Fog                              2760
         Scattered Clouds                 2209
         Partly Cloudy                    2091
         Shallow Fog                      1860
         Mostly Cloudy                    1537
         Light Rain                       1302
         Partial Fog                      1031
         Patches of Fog                    901
         Thunderstorms and Rain            486
         Heavy Fog                         421
         Light Drizzle                     414
         Rain                              394
         Unknown                           383
         Blowing Sand                      378
         Overcast                          326
         Thunderstorm                      192
         Light Thunderstorms and Rain      176
         Drizzle                           112
         Light Fog                          64
         Light Thunderstorm                 64
         Heavy Rain                         28
         Heavy Thunderstorms and Rain       22
         Thunderstorms with Hail            11
         Light Sandstorm                     6
         Squalls                             6
         Light Rain Showers                  5
         Volcanic Ash                        4
         Light Haze                          4
         Sandstorm                           2
         Rain Showers                        2
         Funnel Cloud                        2
         Light Freezing Rain                 1
         Light Hail Showers                  1
         Heavy Thunderstorms with Hail       1
         Name: _conds, dtype: int64
```

Figure 3.5 : Count of unique values for _conds column in Delhi weather data.

#### 3.2.3.4 Scaling

Scaling is an important step to follow before applying a machine learning model. If the input data is not scaled then because of wide range of values which is fed into the model, we might observe that modelling is taking large time, modelling gets difficulty in converging, model results get affected. We will see an apply one of the below scaling methods:

**Normalization:**

Is the scaling technique where we each value is converted to a value between 0 and 1. Normalization is achieved using below formula:

Xnew = (Xold – Xmin) / (Xmax – Xmin)

**Standardization:**

Is the scaling technique where aim is to create a distribution which is centered around the mean of zero, with unit standard deviation. Below is the formula for standard deviation.

Xnew = (Xold – mean) / Standard Deviation

Where Xnew = new value of the variable

Xold = old value of the variable

Xmin = minimum value of the distribution

Xmax = maximum value of the distribution

We will analyze our data set and will try to use either of the scaling technique.

### 3.2.4 EDA (Exploratory data analysis)

We can derive some interesting facts about our data set using EDA. We have listed our columns in previous section. It would be good to see how the pollution level has changed with time in Delhi. Effect of pollution on factors like rainfall level, fog and haze levels will be analyzed. Doing EDA can help us to understand our data set well and give good idea about the domain the data set is belonging to. We will try to find some unique conditions that exists in the data set followed by some time series analysis.

### 3.2.5 Data Sampling

The data set has the datetime_utc column which provide the data and time when the given record/row was measured. This information is present on the hourly basis. Since our aim is to predict the daily temperature we will use sampling in order to convert/combine the hourly measurements into daily basis. Below API is taken from the pandas org documentation reference (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html) We will use this to do the resampling of our data set.

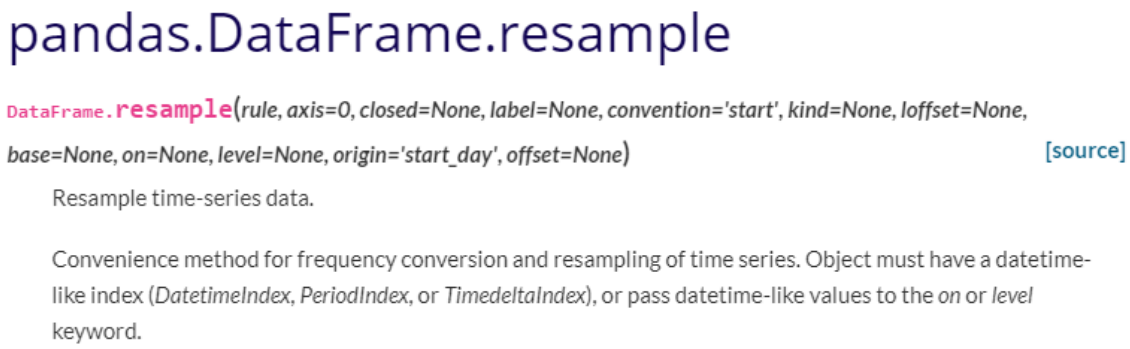



Figure 3 6 : Resample data on daily basis

As we saw in above section in Pandas data frame describe function that we have data time column which is in the object data type. We will convert this to the data time format.

### 3.2.6 Time Series

Time series is sequence of observation which are noted sequentially in time. In time series we are required to predict the future value based upon the previous values provided to the algorithm. One of the major points in time series analyses is at what frequency are the forecast required.

In general, supervised machine learning algorithms needs some input and output to learn from, using which it can output predicted value based upon the unknown input. For our time series data, we need to create this input/output which will be learned by our model.

Below is an example of the approach we will be using in our time series analysis. This shows time series data for a feature X, for time series spanning from 0 to 9. Consider frequency factor of 3 to provide input to our model we will consider previous 3 values (often called as the lag values) for which output will be the fourth value (current value). Using this approach, the table shows the input value for the feature x and output value y which the model should learn.

| Feature X | 20 | 26 | 15 | 19 | 30 | 45 | 32 | 21 | 12 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|
| Time unit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

| Input X | Output Y |
|---|---|
| [20, 26, 15] | 19 |
| [26,15,19] | 30 |
| [15,19,30] | 45 |
| [19,30,45] | 32 |
| [30,45,32] | 21 |
| [45,32,21] | 12 |
| [32,21,12] | 19 |

Figure 3.7 : Time series generation.

We will be using keras Python API during implementation to perform this time series generation. Below is the detail of the API captured from the documentation of keras (https://keras.io/api/preprocessing/timeseries/)

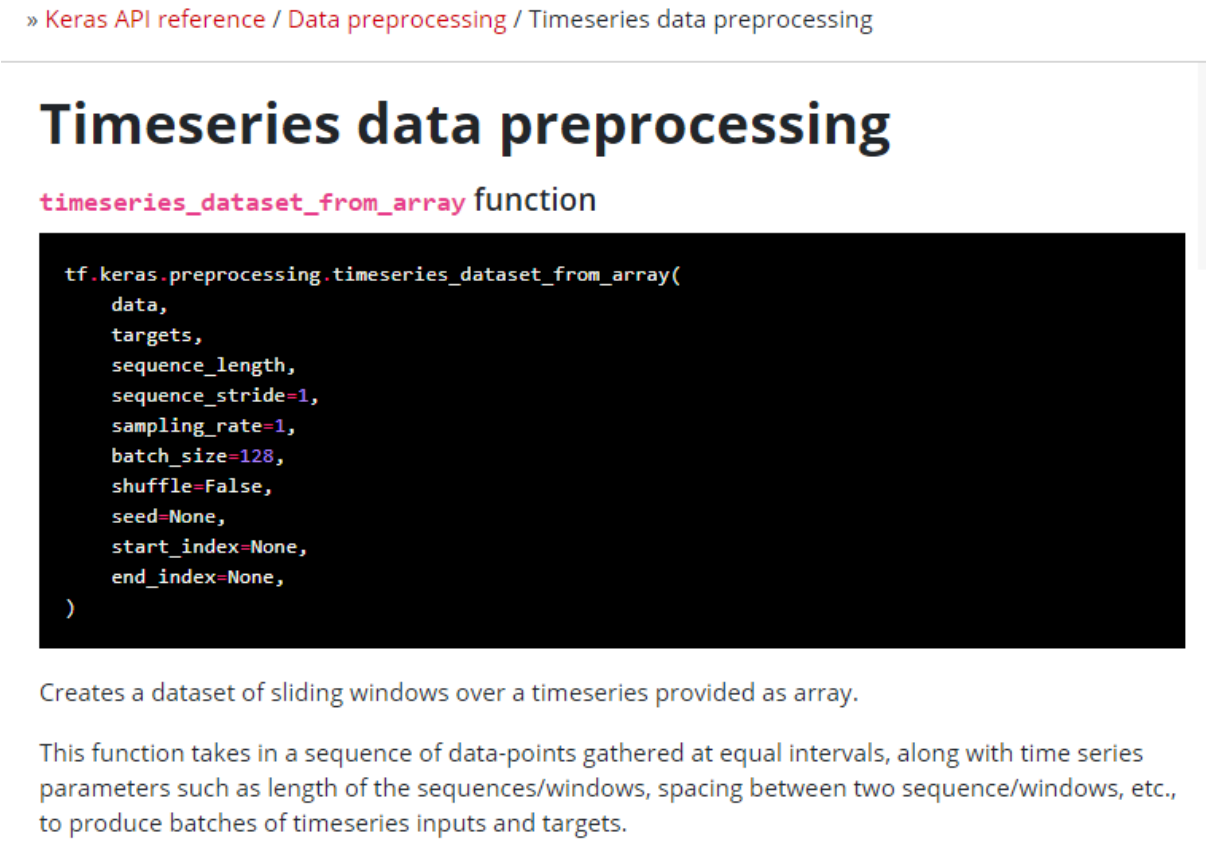



Figure 3.8 : Time series generation using Keras API.

### 3.3 Evaluation Metrics

We will be using one or more of the below evaluation metrics for the interpretation of the results for time series forecast of temperature on our data set.

1.) **Mean Absolute Error:** Is calculated using the mean of the absolute error values. Error values are calculated as the difference between the predicted and actual value of the temperature for all samples or observations.

$$MAE = \frac{\sum(i = 1\ to\ n)|yi\ predicted - yi\ actual|}{\text{n}}$$

Where n is the total number of inputs to the algorithm for which temperature (y) is predicted.

2.) **Mean Square error and Root Means Square error:** Is calculated as the mean value for the square of error term for complete sample output, whereas root mean square error is just the square root of the means square error. Error is calculated after summing the difference between the predicted and the actual temperature values for all samples/inputs.

$$MSE = \frac{\sum(i = 1\ to\ n)\ (yi\ predicted - yi\ actual)^2}{\text{n}}$$

Where n is the total number of inputs to the algorithm for which temperature (y) is predicted.

$$\text{R}MSE = \sqrt{MSE}$$

3.) **Mean Absolute percentage error:** Is calculated as the percentage mean of the absolute error calculated in point 1. Above two error are often dependent upon the scale of the variable. This is good error metrics which is independent of the scale of the variable.

$$MAPE = \frac{1}{\text{n}}\sum(i = 1\ to\ n)\frac{|yi\ predicted - yi\ actual|}{|Yi\ predicted|}$$

Where n is the total number of inputs to the algorithm for which forecast is done for temperature (y).

Below paper (Maiti and Bidinger, 1981) provides good details of these and other performance metrics. It also shows some of the other metrics like Normalized mean square error, various methods for normalization are discussed. It also brings forward relative error measurements.

## 3.4 Proposed Methods

Here we will discuss the models we want to use on our data set to do the time series prediction. We will use below models to perform prediction of the temperature on the Delhi data set.

### 3.4.1 Neural Network

These are amongst the most significant and frequently used Machine Learning algorithms. They have been used in a variety of fields and, when properly trained, are very durable. This model is inspired by human brain and can be used to analyze time series data.
Long short-term memory (LSTM) is a type of Recurrent neural network, which we will be using in our implementation. LSTM remembers values from past observation which can be helpful in time series prediction. It also helps to counter the vanishing gradient problem in Neural Networks.
We will be using Keras library to implement the Neural Network below is the main API we would be using (https://keras.io/api/layers/recurrent_layers/lstm/). As shown in the below API one of the main parameter input is the return_sequences. If set to true a LSTM cell will output one hidden output for each time stamp. This is very useful if we want to stack the output of the LSTM layer, such that the subsequent LSTM layer has multidimensional input. We will use this parameter in our model generation and will try to tune it to get the best results.

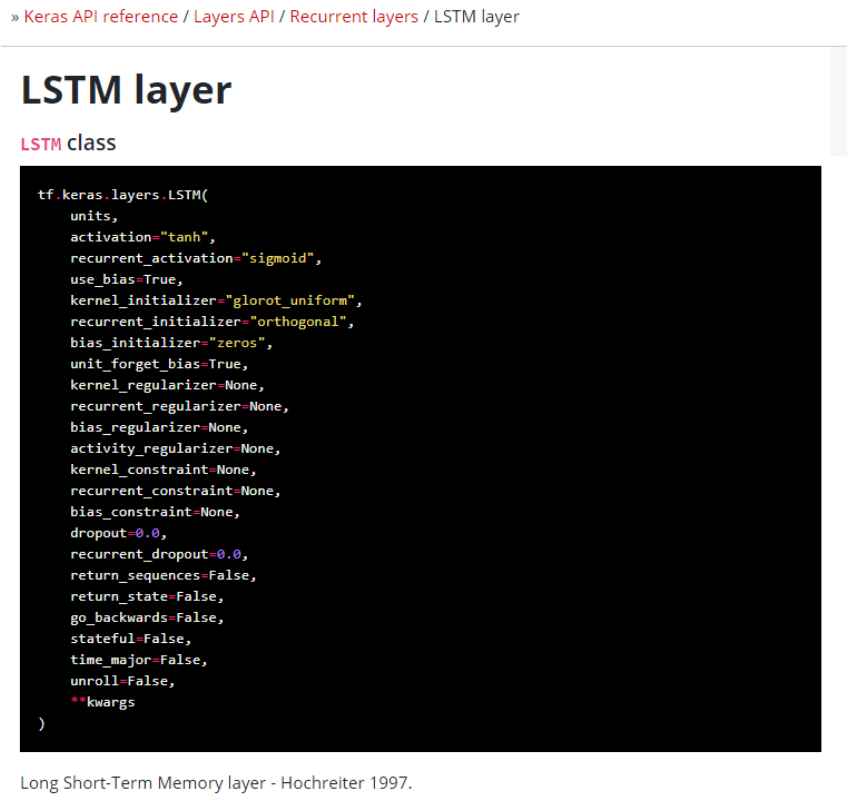



Figure 3.9 : LSTM layer Keras API.

Below diagram shows a pictorial view of return_sequences parameter when set as true where output is generated for each time stamp:

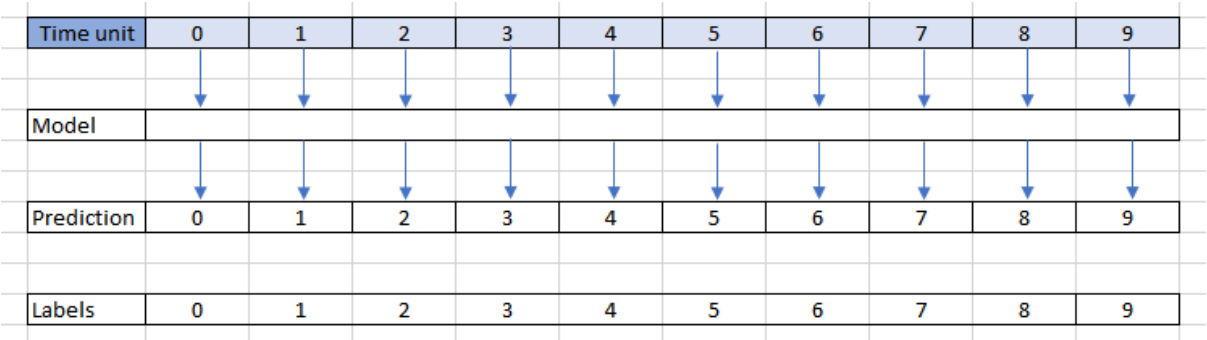



Figure 3.10 : Understanding return sequences parameter when set to true.

Below diagram shows a pictorial view of return_sequences parameter when set as false where output is generated for the last time stamp:

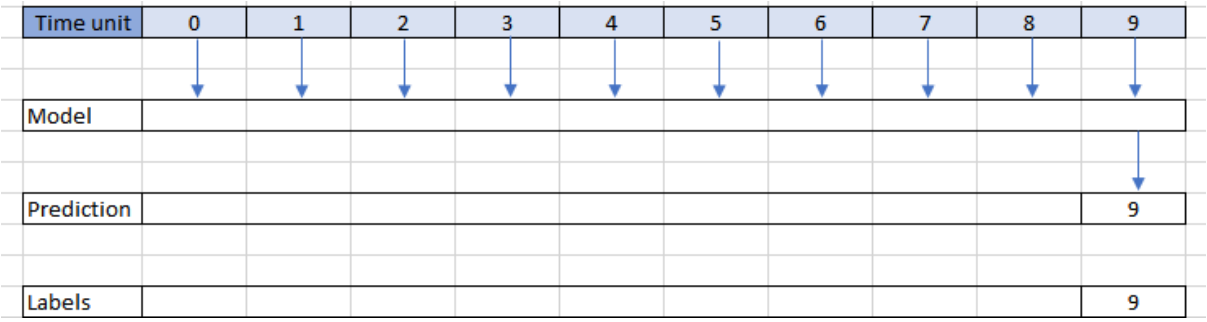



Figure 3.11 : Understanding return sequences parameter when set to false.

Similarly, we would be exploring other Neural network parameters like activation function. Activation is one of the most important inputs as it governs the amount of non-linearity in the model. Good activation function can help to learn more complex tasks.
We will try various activation function below and will use the one which is provides the best results.

**Tanh:** Nonlinear Tangent function which ranges from –1 to +1.

**Relu:** Rectified linear function which ranges from 0 to infinity.

y = 0 if x <= 0

y = x otherwise

**Sigmoid:** It is a function with shape of S which ranges from 0 to 1.

$$y = \frac{1}{1 + e^{-x}}$$

We will also try to tune our model by adding different Dense layer, dense layer is fully connected which means that each neuron of the previous layer is connected to dense layer. The dimension of the output can be controlled by the added dense layer. Below is the API we will use from the keras for this (https://keras.io/api/layers/core_layers/dense/)

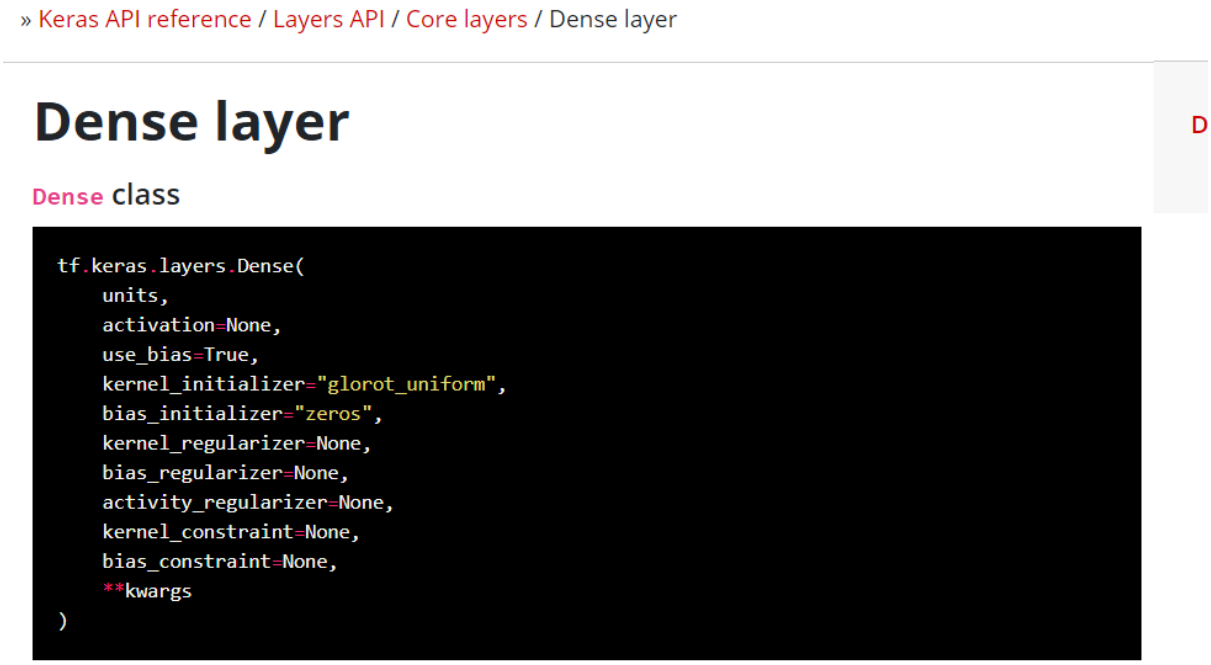



Figure 3 12 Keras Dense layer  API.

We will also analyze during implementation adding the effect of 1 dimension convolution layer in the model along with LSTM and see the model behaviour. Maxpooling should be used along with the convolution layer. Convolution layer applies a filter and outputs a feature map after applying this filter. We can apply the filter on a time series of length t versus number of variables in the time series. Our filter dimension should match the number of variables in the time sereis. We can use this kernal to perform the convolution. Output will depend upon number of kernel we will be using.

Below is a pictorial diagram showing the movement of the filter over the time series using a 1 dimensional convolution network.

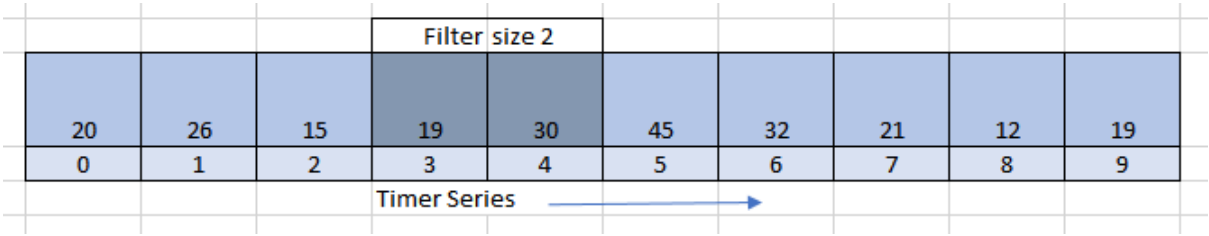



Figure 3.13 : 1 dimensional Convolution neural network for time series.

We will be further applying the MaxPooling layers which will be reducing the output of the convolution 1D layer. Factor by which pooling needs to be done will be tuned during implementation.

### 3.4.2 ARIMA

ARIMA is an abbreviation for autoregressive integrated moving average model. It considers the lagged observation in the time series data and uses a relationship between an observation and the lagged ones. It also uses the difference of the time series with the previous observation. This is done in order to make the time series stationary. This model uses the concept of moving average. Moving average creates different subset of complete data and for each subset it creates an average value. Length of the subset doesn't change for moving average. Suppose we had temperature for 5 days when the temperature value for $6^{th}$ day comes we don't increase the size to 6 but the oldest day value moves out and the recent value is incorporated. ARIMA model is made of three components:

1.) AR(Autoregression) which is often given notation of (p). This represents the lag factor which denotes the number of lags we want to use in the models.
2.) I (Integrated) which is often given notation of (d). This represents number of times we want to do differencing.
3.) MA (Moving Average) which is often given notation of (q). This represents the width of the moving average window we want to use.

ARIMA model is specified with these three parameters (p, d, q).
If integration function which is the differencing operation is not needed to be performed we can set the value of d as zero. This will make model equivalent to model ARMA. We will analyze the data set will try to come up with these three parameters such that our model results are good.

**Autocorrelation and Partial Autocorrelation Function:**

This is a function in time series data which plots the correlation between the different parts of time series. We can term it as self-correlation but at different chunk of time steps. It takes a lagging factor into account and then looks into the past to estimate the likeness. Similarly,

Partial Autocorrelation function for time series data uses the partial correlation coefficients w.r.t to the lagging observations. Partial correlation considers the correlation at different lag observations in past often referred that it considers all lower order lags.
We will draw the autocorrelation function and based upon the plot will try to come up with the value of AR (p) which should be used for our model.
Similarly, we will draw the partial auto correlation function and based upon the plot will try to come up with the value of MA (q) which should be used for our model.
Below are the main points which we will analyze and try to answer from the partial autocorrelation and autocorrelation function plots during implementation:

1.) ACF or PACF plot shows tilting towards zero value.
2.) ACF or PACF plot shows a threshold value after which the value turns to be zero.
3.) There is no tilting value remain close to one.
4.) Both ACF and PACF plots tilts toward zero.
5.) Check and analyse upper and lower confidence level.

**Checking Stationarity:**

We will check the seasonality in the data set which is referred as the change in temperature which happens within regular intervals. If our temperature time series exhibit any seasonality then we need to make it stationary. We would need to do differencing in order to make the time series stationary. It is kind of iterative process where we do a differencing and then check for stationarity, if it still exhibits any seasonality then we would repeat the process of differencing again. Below are few ways we will use in implementation to check this:

1.) We will try to see the mean and standard deviation by matplot plots and try to see if the mean and standard deviation remains constant or not.
2.) We will use augmented dickey fuller test: It basically uses null hypothesis to check if time series is stationary or not. We will use below python library to execute the dickey fuller which is “adfuller” present in package statsmodels.tsa.stattools.

Based upon stationarity we will be able to determine Integration parameter (d) for ARIMA. We will use below library function in python to perform ARIM modelling on our data set. (https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html)

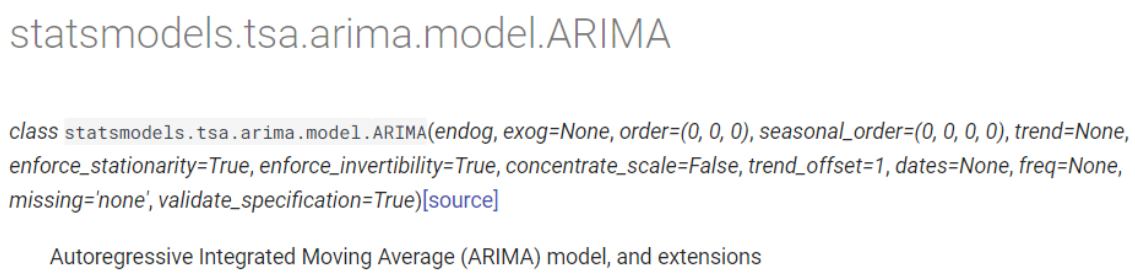



Figure 3.14 : Stats model ARIMA API.

## 3.5 Summary

We have discussed in this section the details of the Research methodology we would like to use. We highlighted the libraries and preprocessing steps and preprocessing we would be following for our work on the Delhi time series data. There are different metrics available for evaluating the time series forecast. We have discussed these metrics and the one which we would like to follow in our implementation. We went through the proposed modelling methods we would be using for our study. Further we went through the details of the models and how we would approach the model building steps. We went through the modelling parameters which are very important, and we should be following closely and tuning in order to have good evaluation results.

# CHAPTER 4: ANALYSIS

## 4.1 Introduction

In this chapter we will perform the analysis and implementation of the various approach towards the Delhi weather data set we have to perform time series prediction. We will start with understanding of our data set and perform handling of missing values, categorical variables and then proceed with processing of outliers. We will than go further and try to perform analysis of various columns in our data set. We will try to get some meaningful information using visualization and graphical plotting to see how various features effect the environment in Delhi region. After this we will perform the sampling of the data set which is required for the modelling procedure. After sampling we will go through the splitting of data in test and train set followed by generation of time series data used by Neural Network modelling. We will also see how this is going to be different from the ARIMA model which uses the lag and moving window parameter for training. Finally, we will go through detail implementation of our chosen models. We will be implementing Neural networks and ARIMA. In details we will see various hyper parameters which we have tuned and the final values of these hyper parameters we have used for model training and prediction. In detail tuning of model structures and derivation of model tuning parameters will be discussed in this section.

## 4.2 Performing Data Preprocessing

In this session we will perform the various preprocessing steps which we had discussed in section 3.2.3. We brough forward in this section the approach we will be using on the data processing. In this session we will further analyze and perform the data preprocessing which will include cleaning and making it fit for feeding into the model.

### 4.2.1 Treating Missing values

We will follow an approach to remove all the columns which have missing values greater than 50 in our data set. Because they would not be helpful in analysis and modelling procedure. Figure 4.1 shows the removal of the columns: [_heatindexm, _precipm, _wgustm, _windchillm] which has null/missing values greater than this threshold.

In [343]:

```
# Remove columns which have missing vlaues greater than 50 %
removingThreshold = 50

missingPer = round(100*(dataDf.isnull().sum()/len(dataDf.index)), 2)

# Remove columns which have mostly nan values/greater than set threshold
for colName, per in missingPer.iteritems():
    if (per >= 50):
        print ("Removing column ", colName, " missing/nan percentage ", per)
        dataDf = dataDf.drop(colName, axis=1)

print()
print(dataDf.info())
```

```
Removing column  _heatindexm  missing/nan percentage  71.13
Removing column  _precipm  missing/nan percentage  100.0
Removing column  _wgustm  missing/nan percentage  98.94
Removing column  _windchillm  missing/nan percentage  99.43

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100990 entries, 0 to 100989
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   datetime_utc  100990 non-null  object
 1   _conds        100918 non-null  object
 2   _dewptm       100369 non-null  float64
 3   _fog          100990 non-null  int64
 4   _hail         100990 non-null  int64
 5   _hum          100233 non-null  float64
 6   _pressurem    100758 non-null  float64
 7   _rain         100990 non-null  int64
 8   _snow         100990 non-null  int64
 9   _tempm        100317 non-null  float64
 10  _thunder      100990 non-null  int64
 11  _tornado      100990 non-null  int64
 12  _vism         96562 non-null   float64
 13  _wdird        86235 non-null   float64
 14  _wdire        86235 non-null   object
 15  _wspdm        98632 non-null   float64
dtypes: float64(7), int64(6), object(3)
memory usage: 12.3+ MB
```

Figure 4 1 : Removing columns with missing value greater than 50 %

We do not have zero missing values yet. Analyzing the mean/median/mode and standard deviation of the columns we see that would be ok to impute the values with median for numeric and mode for categorical variables. However, we should follow this within certain limit. We have fixed a threshold of 6 percent all the columns who has missing values less than 15 percentage we will impute the missing value with median or mode depending upon the type of variable. Figure 4.2 shows this operation.

In [383]:

```
nanInfo = round(100*(dataDf.isnull().sum()/len(dataDf.index)), 2)
# Threshold in percetage values below this will imputed with median/mode.
ImputeThreshold = 6

for colInfo in nanInfo.iteritems():

    dtype = dataDf[colInfo[0]].dtypes.name
    if colInfo[1] > 0 and colInfo[1] < ImputeThreshold:

        if ((dtype) == 'object'):
            print("Column ",colInfo[0], "type ", dtype, " missing values % ", colInfo[1])
            print ("Imputing ", dtype, " with mode ", dataDf[colInfo[0]].mode()[0])
            dataDf[colInfo[0]].fillna(dataDf[colInfo[0]].mode()[0], inplace = True)
        else:
            print("Column ",colInfo[0], "type ", dtype, " missing values % ", colInfo[1])
            print ("Imputing ", dtype, " with median ", dataDf[colInfo[0]].median())
            dataDf[colInfo[0]].fillna(dataDf[colInfo[0]].median(), inplace = True)

print (round(100*(dataDf.isnull().sum()/len(dataDf.index)), 2))
print ("total records/rows {0} total columns {1}".format(dataDf.shape[0], dataDf.shape[1]))
```

```
Column  _conds type  object  missing values %  0.07
Imputing  object  with mode  Fog
Column  _dewptm type  float64  missing values %  0.61
Imputing  float64  with median  15.0
Column  _hum type  float64  missing values %  0.75
Imputing  float64  with median  59.0
Column  _pressurem type  float64  missing values %  0.23
Imputing  float64  with median  1008.0
Column  _tempm type  float64  missing values %  0.67
Imputing  float64  with median  27.0
Column  _vism type  float64  missing values %  4.38
Imputing  float64  with median  2.0
Column  _wspdm type  float64  missing values %  2.33
Imputing  float64  with median  7.4
datetime_utc     0.00
_conds           0.00
_dewptm          0.00
_fog             0.00
_hail            0.00
_hum             0.00
_pressurem       0.00
_rain            0.00
_snow            0.00
_tempm           0.00
_thunder         0.00
_tornado         0.00
_vism            0.00
_wdird          14.61
_wdire          14.61
_wspdm           0.00
dtype: float64
```

Figure 4 2 : Imputing columns with missing value less than 6 %

We have still two columns left which have missing values of around 14 % since we have more than 1 lakh records, we are dropping rows corresponding to these missing values.

In [406]:

```
dataDf = dataDf[~pd.isnull(dataDf['_wdird'])]
dataDf = dataDf[~pd.isnull(dataDf['_wdire'])]
print (round(100*(dataDf.isnull().sum()/len(dataDf.index)), 2))
```

```
datetime_utc    0.0
_conds          0.0
_dewptm         0.0
_fog            0.0
_hail           0.0
_hum            0.0
_pressurem      0.0
_rain           0.0
_snow           0.0
_tempm          0.0
_thunder        0.0
_tornado        0.0
_vism           0.0
_wdird          0.0
_wdire          0.0
_wspdm          0.0
```

Figure 4 3 : Drop records which have nan values for _wdird, _wdire

### 4.2.2 Handling Categorical Variables

In section 3.2.3.3 we discussed an approach where we want to club the values of categorical variables so that the span of the variables is less. The main reason for this approach is we saw that the variables have less value count for some of the values of categorical variables. Below we are dropping the columns which has biased categorical values as these would not be much useful in analysis and modelling process.

In [429]:
```
print ("value Counts")
print (dataDf['_hail'].astype('category').value_counts())
print (dataDf['_thunder'].astype('category').value_counts())
print (dataDf['_tornado'].astype('category').value_counts())
print (dataDf['_fog'].astype('category').value_counts())
print (dataDf['_snow'].astype('category').value_counts())
print (dataDf['_rain'].astype('category').value_counts())
dataDf = dataDf.drop(['_snow', '_hail', '_thunder', '_tornado', '_rain', '_fog'], axis=1)
```

```
value Counts
0    86224
1       11
Name: _hail, dtype: int64
0    85372
1      863
Name: _thunder, dtype: int64
0    86233
1        2
Name: _tornado, dtype: int64
0    80853
1     5382
Name: _fog, dtype: int64
0    86234
1        1
Name: _snow, dtype: int64
0    83950
1     2285
Name: _rain, dtype: int64
```

Figure 4 4 : Remove values with biased value counts for categorical variables.

For columns _conds, _wdire we have analyzed and clubbed together the values bringing up the value counts for some of the variable values.

In [433]:
```
dataDf['_conds'] = dataDf['_conds'].replace(['Haze', 'Mist', 'Fog', 'Shallow Fog', 'Partial Fog', 'Patches of Fog', \
                                             'Heavy Fog', 'Light Fog', 'Light Haze'], 'Haze/Fog')

dataDf['_conds'] = dataDf['_conds'].replace(['Widespread Dust', 'Blowing Sand', 'Volcanic Ash', 'Sandstorm', \
                                             'Light Sandstorm'], 'Dust')

dataDf['_conds'] = dataDf['_conds'].replace(['Scattered Clouds', 'Partly Cloudy', 'Mostly Cloudy', 'Funnel Cloud', \
                                             'Overcast'], 'Cloudy')

dataDf['_conds'] = dataDf['_conds'].replace(['Light Rain', 'Light Drizzle', 'Rain', 'Drizzle', 'Heavy Rain', \
                                             'Light Rain Showers', 'Rain Showers', 'Light Freezing Rain', \
                                             'Light Hail Showers'], 'Rain')

dataDf['_conds'] = dataDf['_conds'].replace(['Light Thunderstorms and Rain', 'Light Thunderstorm', \
                                             'Heavy Thunderstorms and Rain', 'Thunderstorms with Hail', \
                                             'Heavy Thunderstorms with Hail'], 'Thunderstorm')

dataDf['_conds'] = dataDf['_conds'].replace(['Squalls', 'Unknown', 'Thunderstorms and Rain'], 'Others')
```

In [435]:
```
dataDf['_conds'].astype('category').value_counts()
```

Out[435]:
```
Haze/Fog        51046
Smoke           19839
Cloudy           6138
Clear            3128
Dust             2886
Rain             1946
Others            821
Thunderstorm      431
Name: _conds, dtype: int64
```

Figure 4 5 : Club categorical values for _conds

```
In [423]:
dataDf['_wdire'] = dataDf['_wdire'].replace(['NNW', 'WNW'], 'NW')

dataDf['_wdire'] = dataDf['_wdire'].replace(['NNE', 'ENE'], 'NE')

dataDf['_wdire'] = dataDf['_wdire'].replace(['ESE', 'SSE'], 'SE')

dataDf['_wdire'] = dataDf['_wdire'].replace(['WSW', 'SSW'], 'SW')

dataDf['_wdire'] = dataDf['_wdire'].replace(['Variable'], 'South')

In [424]:
dataDf['_wdire'].astype('category').value_counts()

Out[424]:
North    19034
NW       17277
West     11888
SE       10901
SW       10381
NE        7973
East      7233
South     1548
Name: _wdire, dtype: int64
```

Figure 4 6 : Club Categorical values for _wdire

### 4.2.3 Processing Outliers

We saw in section 3.2.3.2 that there are outliers present in the data set. We have processed the outliers after analyzing the box plot of each of the variables. Quantile is being used to handle the outliers. Different values of Quantile are tried and the box plot were analyzed to get to the optimal value which most accurately qualifies a outlier. Figure 4.7 shows the processing of the outliers on the data set. Figure 4.8 shows the final box plot after outliers are processed.

```
In [40]:
def processOutliers(col, dataDf, uQvalue, lQvalue):
    tmpDf = dataDf
    plt.subplot(121)
    sns.boxplot(tmpDf[col])
    plt.title('Plot before removing Outliers\ntotal rows {0}'.format(len(tmpDf.axes[0])))

    if (uQvalue > 0):
        uBound = tmpDf[col].quantile(uQvalue)
        print(col + " :Number of Prominent Outliers above {0} : {1}, Total rows {2}". \
            format(uBound, len(tmpDf[(uBound <= tmpDf[col])].axes[0]), len(tmpDf.axes[0])))

        # Remove the prominent outliers above the uBound range.
        tmpDf = tmpDf[(uBound > tmpDf[col])]

    if (lQvalue > 0):
        lBound = tmpDf[col].quantile(lQvalue)
        print(col + " :Number of Prominent Outliers below {0} : {1}, Total rows {2}". \
            format(lBound, tmpDf[(lBound >= tmpDf[col])][col].count(), len(tmpDf.axes[0])))

        # Remove the prominent outliers below the lBound range.
        tmpDf = tmpDf[(lBound < tmpDf[col])]
    plt.subplot(122)
    sns.boxplot(tmpDf[col])
    plt.title('Plot after removing Outliers\ntotal rows {0}'.format(len(tmpDf.axes[0])))
    plt.tight_layout()
    plt.show()
    return tmpDf

dataDf = processOutliers('_dewptm', dataDf, 0.999999, 0.0001)
dataDf = processOutliers('_hum', dataDf, 0.9999999, -1)
dataDf = processOutliers('_pressurem', dataDf, 0.99998, -1)
dataDf = processOutliers('_vism', dataDf, 0.99995, -1)
dataDf = processOutliers('_tempm', dataDf, 0.99991, -1)
dataDf = processOutliers('_wspdm', dataDf, 0.99991, -1)
dataDf = processOutliers('_wdird', dataDf, 0.99996, -1)
```

Figure 4 7 : Processing outliers

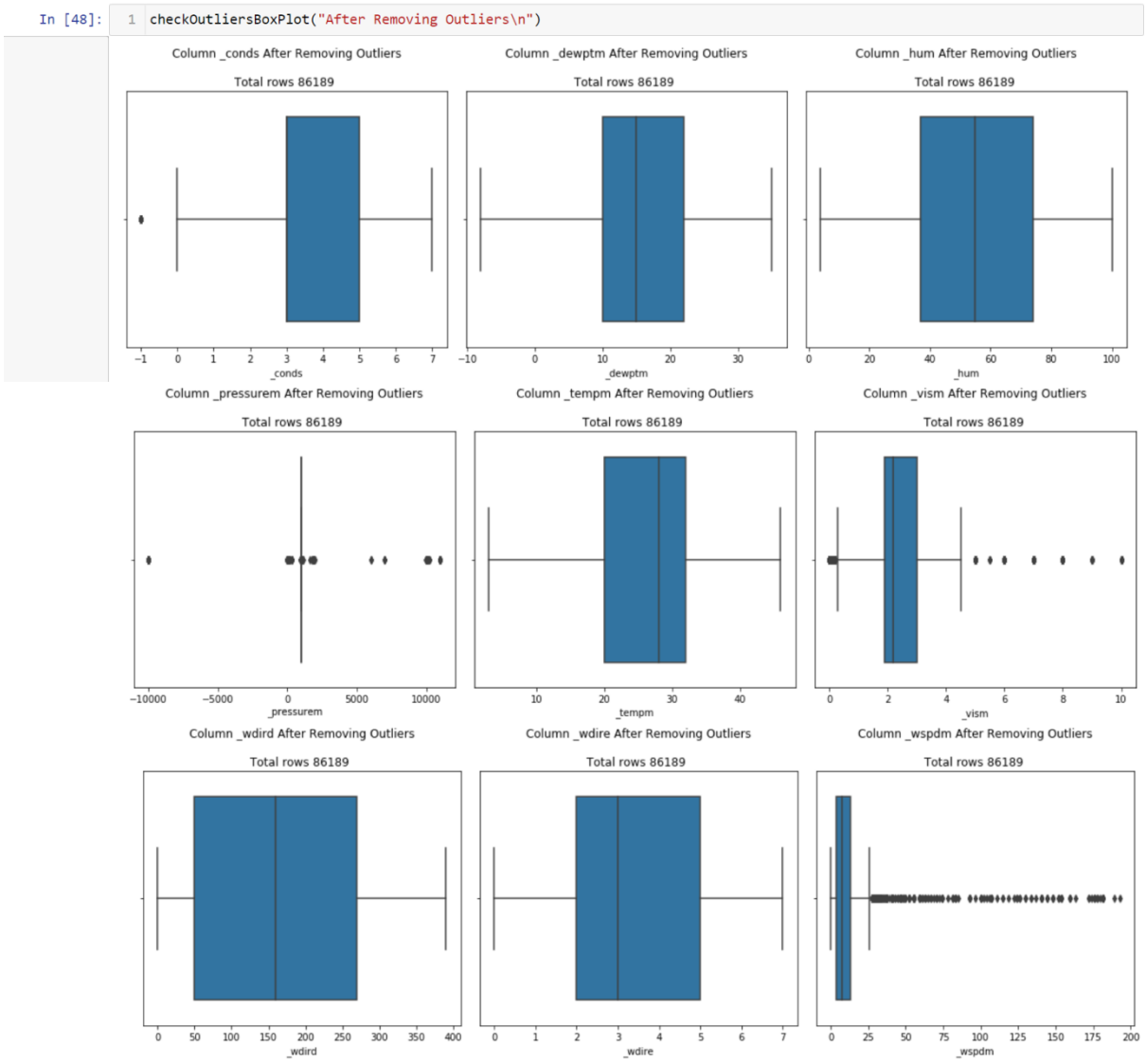



Figure 4 8 : Box plot to visualize outliers

## 4.3 Performing Exploratory data analysis (EDA)

EDA is very important to get the feel of the data set. One direct intuition is to check how the temperature distribution looks like. Figure 4.9 shows the distribution of temperature. We can see that the temperature distribution is high between 25 to 35 degree and falls down at the towards left or right of this range.

```
In [297]:
1 sns.distplot(dataDf['_tempm'])
2 plt.show()
```

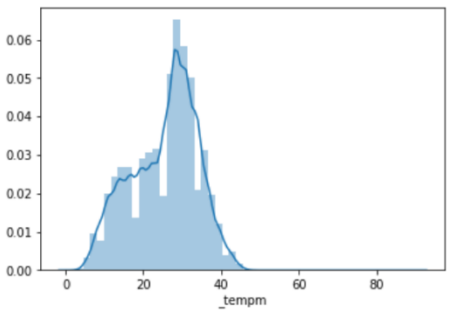



Figure 4 9 : Distribution plot for _tempm

Similar behavior is seen for humidity levels. High around the central range of 45 to 55 and falls towards the left and right of this range.

```
In [296]:
1 sns.distplot(dataDf['_hum'])
2 plt.show()
```

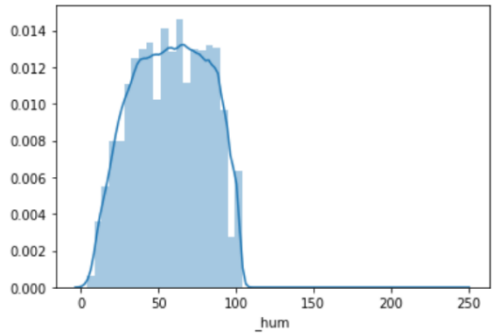



Figure 4 10 : Distribution plot for _hum

Analyzing the weather condition in Delhi we see that mostly it remains filled with haze/fog, followed by smoky and cloudy. Figure 4.11 shows this using bar plot. This shows that there is high level of smoke and haze in the area possible due to pollution and degradation of nature. We also notice that there is very less rainfall in Delhi. Similarly, we checked in which direction the wind flows in the area. Most prevalent direction is North and West, followed by south east and south west.

```
In [438]:
1 dataDf['_conds'].value_counts().plot(kind = 'bar')
2 plt.xlabel("_conds")
3 plt.show()
```

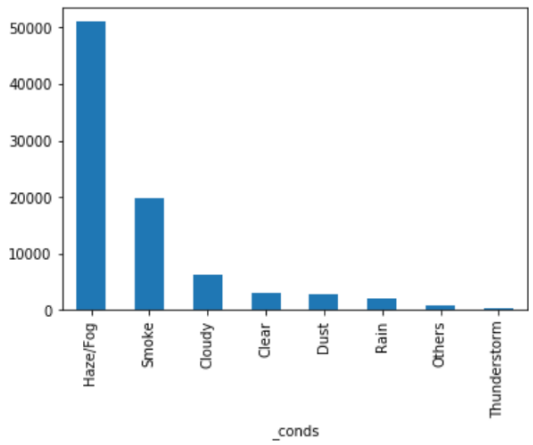



Figure 4 11: Weather condition plot

```
In [439]:
1 dataDf['_wdire'].value_counts().plot(kind = 'bar')
2 plt.xlabel("_conds")
3 plt.show()
```

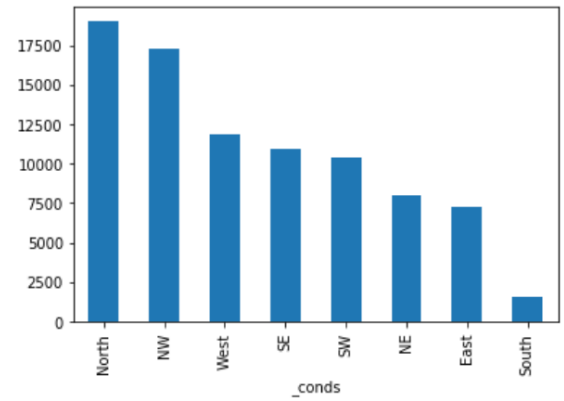



Figure 4 12 : Wind direction plot

Heat plot in figure 4 13 shows the monthly average temperature range across years. We see that for month of November, December, January February and March the temperature is relatively lower in comparison to other months. On the contrary May, June, July are the months when the temperature tends to reach its highest.

In [909]:
```
# Convert the date time utc column to date time object type
dataDfEda["datetime_utc"] = pd.to_datetime(dataDfEda["datetime_utc"])

# Set data frame index to create datetime utc column, which is needed for sampling.
dataDfEda = dataDfEda.set_index("datetime_utc")

# Resample the data frame monthly and get the average temprature for a month
dataDfEda = dataDfEda.resample("M").mean()
dataDfEda = dataDfEda.fillna(dataDf.median())

# Create year and month column
dataDfEda["year"] = dataDfEda.index.year
dataDfEda["month"] = dataDfEda.index.month
```

In [910]:
```
plt.figure(figsize=(20,8))
ym_temp = dataDfEda.pivot("year", "month", "_tempm")

sns.heatmap(ym_temp, annot=True, cmap='coolwarm')
plt.show()
```

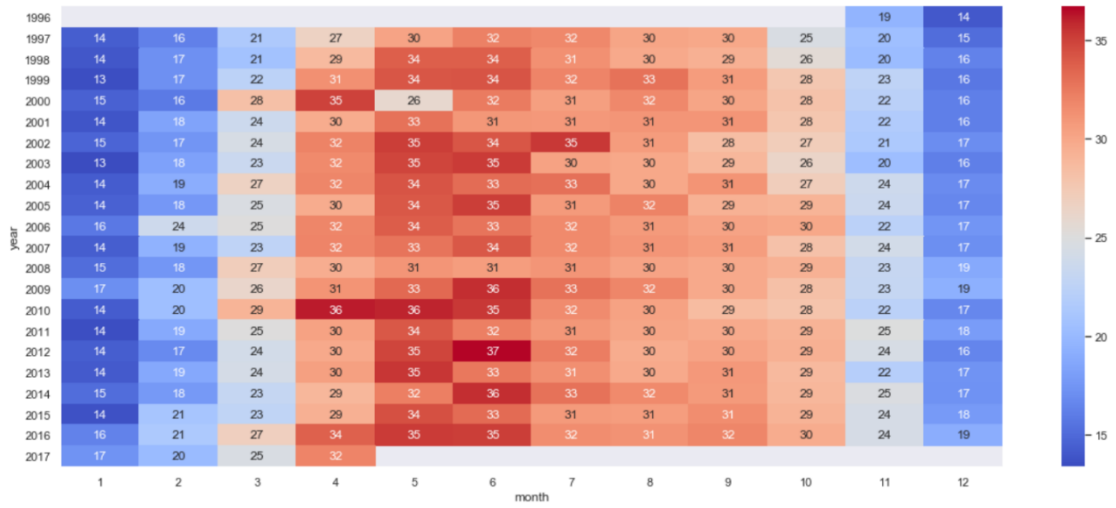



Figure 4 13 : Plot showing monthly temperature per year

## 4.4 Perform Data Sampling

Since our aim is to do a time series prediction of daily temperature and we see in the data set temperature is provided per hour. We will perform the sampling which will roll up the daily temperature and output an average temperature per day. Figure 4 14 performs this operation to get a data frame having daily temperature.

In [1051]:
```
# Convert the data time utc column to data time object type
dataDf["datetime_utc"] = pd.to_datetime(dataDf["datetime_utc"])

# Set the index to the data time object, this is needed to perform sampling
dataDf = dataDf.set_index("datetime_utc")

# Perform daily sampling by taking mean of temprature during day.
dataDf = dataDf.resample("D").mean()
```

In [1052]:
```
dataDf = dataDf.fillna(dataDf.median())
dataDf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 7480 entries, 1996-11-01 to 2017-04-24
Freq: D
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   _conds      7480 non-null   float64
 1   _dewptm     7480 non-null   float64
 2   _hum        7480 non-null   float64
 3   _pressurem  7480 non-null   float64
 4   _tempm      7480 non-null   float64
 5   _vism       7480 non-null   float64
 6   _wdird      7480 non-null   float64
 7   _wdire      7480 non-null   float64
 8   _wspdm      7480 non-null   float64
dtypes: float64(9)
memory usage: 584.4 KB
```

Figure 4 14 : Sampling of data to daily temperature

From this we further get temperature data frame which will be used during modelling. Figure 4 15 shows the variation of the temperature with time.

In [76]:
```
tempDataDf = pd.DataFrame(list(dataDf['_tempm']), columns=['temprature'])
plt.figure(figsize=(20,8))
plt.plot(tempDataDf)
plt.xlabel("Time series")
plt.ylabel("Temprature")
plt.title("Delhi Daily Temprature")
plt.show()
```

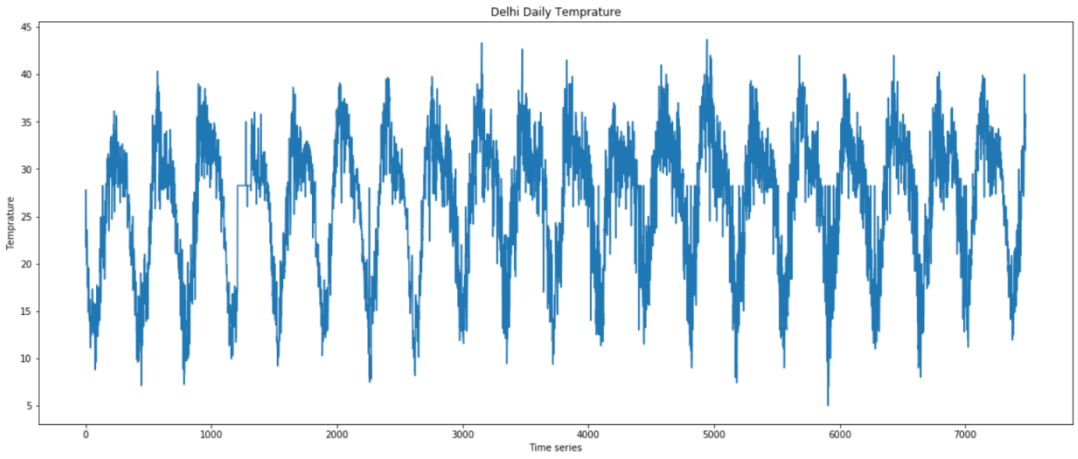



Figure 4 15 : Delhi daily temperature variation

## 4.5 Data splitting and Scaling

We would be splitting our data into train and test which will be used during modelling phase. Train data will be input to the modelling and will be used for model training and building. We will then see how our model is performing on the testing data set. Figure 4 16 shows this operation.

```
In [1139]:
from sklearn.model_selection import train_test_split

trainDf, testDf = train_test_split(tempDataDf, train_size=0.9, test_size=0.1, random_state = 100)

print ("test shape {0}, train shape {1}".format (testDf.shape, trainDf.shape))
```

```
test shape (748, 1), train shape (6732, 1)
```

```
In [1140]:
trainDf.head()
```

Out[1140]:

| | temprature |
|---|---|
| **5307** | 35.666667 |
| **2022** | 36.291667 |
| **6770** | 32.750000 |
| **1059** | 33.130435 |
| **5277** | 34.000000 |

Figure 4 16 : Train and test data split

Further it is essential to perform scaling which helps during modelling phase. We have performed Min Max scaling on the data set which will convert our values of temperature between 0 and 1. Figure 4 17 shows the scaling step.

```
In [1141]:
scaler = MinMaxScaler()
trainData = scaler.fit_transform(trainDf)

print ("trainData scaled size {0}".format (trainData.shape))
trainData[0:10]
```

```
trainData scaled size (6732, 1)
```

```
Out[1141]:
array([[0.79310345],
       [0.80926724],
       [0.71767241],
       [0.72751124],
       [0.75      ],
       [0.33103448],
       [0.60239697],
       [0.59365204],
       [0.69827586],
       [0.6683908 ]])
```

```
In [1142]:
testData = scaler.transform(testDf)

print ("testData scaled size {0}".format (testData.shape))
testData[0:10]
```

```
testData scaled size (748, 1)
```

```
Out[1142]:
array([[0.49137931],
       [0.35344828],
       [0.77586207],
       [0.72306034],
       [0.45258621],
       [0.2112069 ],
```

Figure 4 17 : Scaling of data

We have performed 90 % train to 10 % test data split. We should make sure that test data should not be involved at all during the training procedure that's why we performed only scaler transform on test data while doing fit and transform on train data.

For Arima model we are directly splitting the data in train and test as show in figure 4 18.

```
In [166]:
trainDataSize = 6732
trainDf = tempDataDf[:trainDataSize]
testDf = tempDataDf[trainDataSize:]

(6732, 1)
(748, 1)
```

Figure 4 18 Train and test data split for Arima

## 4.6 Time Series data generation

As discussed in section 3.2.6 for time series analysis we need to generate time series data. We will be generating a time series data for 30 days which means our modelling will be using the previous 30 days temperature to predict the temperature for following day. Figure 4 18 shows the generation of time series data for train and test data. We have input data which is of dimension (30, 1) and one output variable. For training we have 6702 records while we have kept 718 records for testing purpose.
For ARIMA modelling we will use first 6732 days as training data (using appropriate lag and moving window values) and then test on the rest 748 days of the data set.

```
In [110]:
from keras.preprocessing.sequence import TimeseriesGenerator
def getTimeSeriesData(tempData, features):
    timeSeries = TimeseriesGenerator(tempData, tempData, length = features, batch_size = len(tempData))
    x_data, y_data = timeSeries[0]
    print ("shape of time series data x ", x_data.shape, " y ", y_data.shape)
    return x_data, y_data

In [111]:
x_data_train, y_data_train = getTimeSeriesData(trainData, 30)

shape of time series data x  (6702, 30, 1)  y  (6702, 1)

In [112]:
x_data_test, y_data_test = getTimeSeriesData(testData, 30)

shape of time series data x  (718, 30, 1)  y  (718, 1)
```

Figure 4 19 : Time series data generation

## 4.7 Neural Network Modelling

Figure 4 19 shows the structure of the Neural Network model which is implemented. As discussed in section 3.4.1. We have made use of CNN 1 dimensional layers and LSTM to do the time series prediction on training data set.

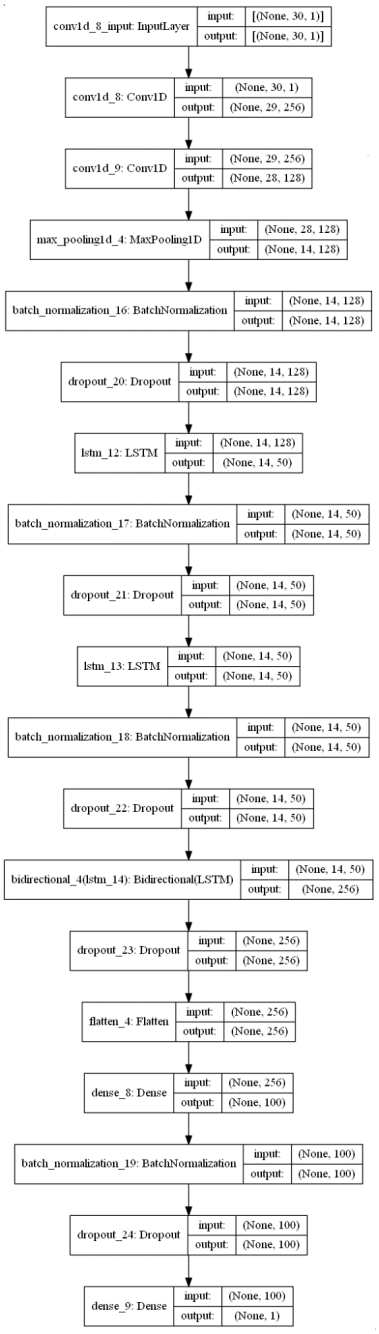



Figure 4 20 : Model Structure

Figure 4 20 shows the model implementation and the model summary with details of number of trainable parameters using python keras API's. We will go through the structure of the model elements next.

In [56]:

```
LR = ReduceLROnPlateau(monitor = 'loss', factor = 0.5, patience = 2, verbose = 1, mode = 'min', \
                       cooldown=0, min_lr=0.0)

filepath = 'models/best_model.hdf5'
checkpoint = ModelCheckpoint(filepath, monitor = 'loss', verbose = 1, save_best_only = True, mode = 'max')

callbacks_list = [LR, checkpoint]

model = Sequential()
model.add(Conv1D(filters = 256, kernel_size = 2, activation = 'relu', input_shape = (30,1)))

model.add(Conv1D(filters = 128, kernel_size = 2, activation = 'relu'))
model.add(MaxPooling1D(pool_size = 2))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(LSTM(units = 50, return_sequences = True, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences = True, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Bidirectional(LSTM(128, activation = 'relu')))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(100, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(1))
model.compile(loss = 'mse', optimizer = keras.optimizers.Adam())
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_8 (Conv1D)            (None, 29, 256)           768
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 28, 128)           65664
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 14, 128)           0
_________________________________________________________________
batch_normalization_16 (Batc (None, 14, 128)           512
_________________________________________________________________
dropout_20 (Dropout)         (None, 14, 128)           0
_________________________________________________________________
lstm_12 (LSTM)               (None, 14, 50)            35800
_________________________________________________________________
batch_normalization_17 (Batc (None, 14, 50)            200
_________________________________________________________________
dropout_21 (Dropout)         (None, 14, 50)            0
_________________________________________________________________
lstm_13 (LSTM)               (None, 14, 50)            20200
_________________________________________________________________
batch_normalization_18 (Batc (None, 14, 50)            200
_________________________________________________________________
dropout_22 (Dropout)         (None, 14, 50)            0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 256)               183296
_________________________________________________________________
dropout_23 (Dropout)         (None, 256)               0
_________________________________________________________________
flatten_4 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 100)               25700
_________________________________________________________________
batch_normalization_19 (Batc (None, 100)               400
_________________________________________________________________
dropout_24 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 101
=================================================================
Total params: 332,841
Trainable params: 332,185
Non-trainable params: 656
_________________________________________________________________
```

Figure 4 21 : Model Summary

### 4.7.1 Convolution Layer 1D

We have stacked two convolution layers in the model which are of 1 dimension. For 1D layer the filter should also be in 1 dimensional. For the first layer we have used the filter of size 2. It derives 256 features out of the input which is of size (30,1). Output can be calculated using below formula:

(Input size + 2 * Padding size – Filter Size) / Stride + 1

For 1st layer input size is 30 with 0 padding size, 2 filter size and stride of 1 output is of dimension (29, 256) for 256 feature maps/filter. Total trainable parameters are computed as total weights plus total bias which is:

256 * 2 + 256 = 768

For 2nd layer input size is 29 with 0 padding size, 2 filter size and stride of 1 output is of dimension (28, 128) for 128 feature maps. Total trainable parameters are computed as total weights plus total bias for this layer which is:

256 * 128 * 2 + 128 = 65664

Relu Activation function was found to be tune in best for the model.

### 4.7.2 LSTM Layer

Three LSTM layer is used in the model. LSTM layers are added after 2nd Convolution 1D layer. First LSTM layer has units (Hidden and Cell State vector dimension) of 50. With 50 units and return sequences set as True the output of the LSTM layer is (14, 50). With input dimension of 14 with 128 features the total number of trainable parameters for the layer is:

4 * ((128 + 50) * 50 + 50) = 35,800.

Second LSTM layer has same units as First layer which are 50. The output of this LSMT layer is (14, 50) with return sequences set as true. With input dimension of 14 with 50 features the total number of trainable parameters for the layer is :

4 *((50 + 50) * 50 + 50) = 20,200.

The Third and the last layer is Bidirectional LSTM layer, we have used the default merge mode as ‘concat’ which concatenates the forward and backward output before going to the next layer.

The input is of dimension (14, 50). With 128 units and return sequences set as False the output of this layer is 128 * 2 = 256. With input dimension of 14 with 50 features the total number of trainable parameters for the layer is:

2 * 4 * ((50 + 128) * 128 + 128) = 183,296.
All three LSTM layers uses Relu activation function. We did tunning with different values of input LSTM Units and parameters like Return sequences, Merge mode and found above settings to be optimal.

### 4.7.3 Max Pooling Layer

Max pooling reduces the size of the feature map with number of features in the map remaining the same. We have used one 1 dimensional max pooling layer of pool size as 2 after 2nd 1 dimensional convolution layer. Then input to the max pooling layer is (28, 128) and with pool size of 2 it reduces the dimension to (14, 128). Please not this layer does not have any trainable parameters.

### 4.7.4 Batch Normalization and Dropouts

As a regularization technique we have used batch normalization and dropouts. We have added the batch normalization after 2nd convolution layer which has output of 128, therefore number of added trainable parameters after batch normalization are 4 * 128 = 512. Another is added after two LSTM layers with 200 (4 * 50) trainable parameters each. Last batch normalization with 400 trainable parameters is added after the dense layer of output 100. We did find out that Batch normalization can help to train the network better and more stably. We also used dropouts after batch normalization which really helps in model not over fitting. We have tried to tune the various values of dropouts between 0.1 to 0.5 and found 0.2 and 0.25 giving good results for our model. Please note dropouts does not add any trainable parameters.

### 4.7.5 Flatten and Dense layer

We have used a flatten layer after bidirectional LSTM layer. It does not add any trainable parameters and just unrolls the output of the LSTM layer. After flatten layer we have used a dense layer of 100 neurons. Trainable parameters for this layer output are:
256 * 100 + 100 = 25700. This layer uses Relu activation function.

Finally, this dense layer is connected to a dense layer with 1 neuron. This happens to be our last layer and gives final output value from the model. Trainable parameters for this layer are:
100 * 1 + 1 = 101. This layer uses linear activation function.

### 4.7.6 Model Callbacks

Learning rate is one of the most vital parameters in the modelling. We have use keras callback to optimize our learning rate when model is training. We have used loss as our measurement metrics which is getting monitored when the model is training, when this metrics stop improving for 2 (patience) epochs learning rate will be reduced by factor of 0.5. We are not using any cool down and allow learning rate to decrease to very low values.

## 4.8 ARIMA

In this section we will go through the building of Arima model. We will go through the derivation of parameters needed to run the Arima model and will also run the Arima model on the temperature data set.

### 4.8.1 Stationarity

Series is said to be Stationary if the mean is constant over time that is if we split the data in multiple parts means should be round about same. It should also have a constant variance and covariance which should be fixed over time within fixed lag. There are many methods to check stationarity we have first drawn the rolling mean to get the feel of the mean values and standard deviation (we took the rolling mean of 12 entries) and then run the augmented dickey fuller test to check the stationarity. Figure 4 22 shows the rolling mean graph for the temperature data set.

In [199]:
```
def drawRollingMean(df):

    plt.figure(figsize = (22, 14))
    # Rolling statistics
    rolling_mean = df.rolling(window=12).mean()
    rolling_std = df.rolling(window=12).std()

    # Rolling statistics plot
    original = plt.plot(df, color = 'blue', label = 'Original Temprature')
    mean = plt.plot(rolling_mean, color = 'yellow', label = 'Rolling Mean')
    std = plt.plot(rolling_std, color='black', label = 'Rolling Std')
    plt.title('Rolling Mean & Standard Deviation')
    plt.legend()
    plt.show()
```

In [200]:
```
rolling_mean = trainDf.rolling(window=12).mean()
tempDataDf_minus_mean = trainDf - rolling_mean
tempDataDf_minus_mean.dropna(inplace=True)
drawRollingMean(tempDataDf_minus_mean)
```

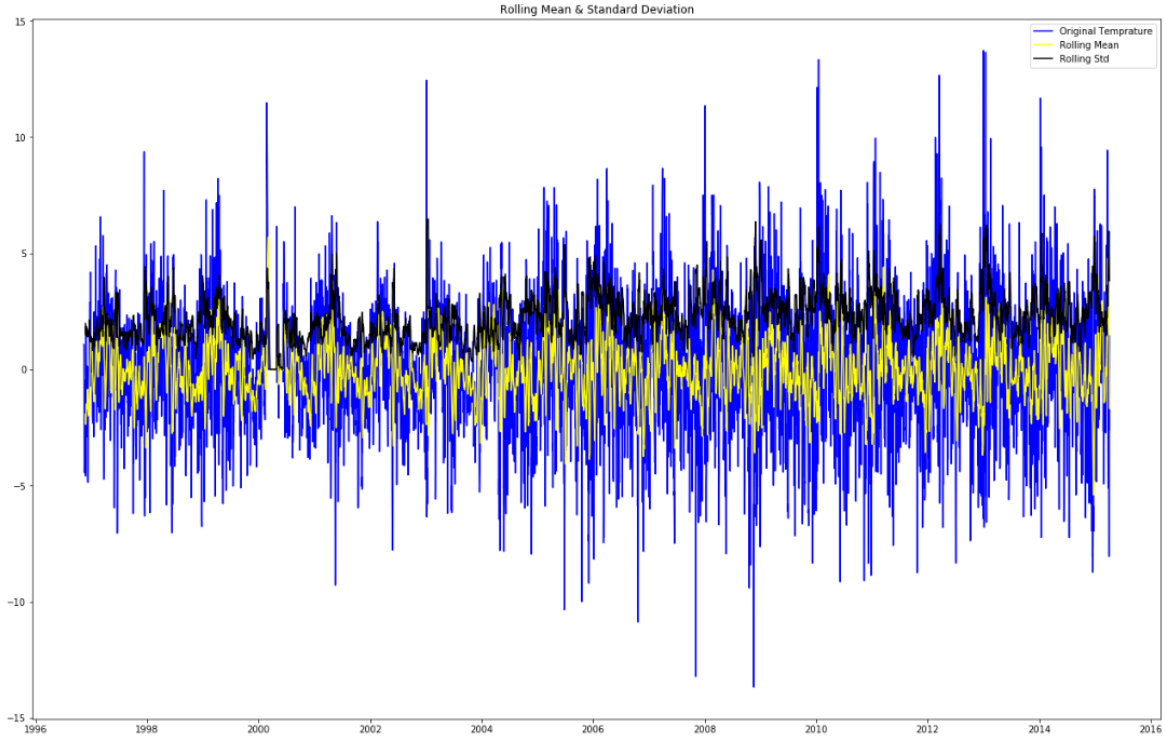



Figure 4 22 : Rolling mean and standard deviation.

Augmented Dickey fuller test has Null hypothesis that time series input is not stationary. Figure 4 23 shows the output of the Augmented Dickey fuller test from the output we can see that for significance level of 5 percent, p values is less than 0.05. Therefore, we can reject the Null hypothesis thus the time series is stationary. Another thing to notice is that the adf statistics is less than the critical value indicating that time series is stationary. This indicates that we don't need to do any differencing (which is used to make time series stationary). Thus, our d parameter during Arima modelling will be zero.

```
In [167]:
from statsmodels.tsa.stattools import adfuller

adf_test = adfuller(trainDf['_tempm'])

#
print ("ADF statistics", adf_test[0])
print ("Critical value @ 0.05 ", adf_test[4]['5%'])
print ('p-Value ', adf_test[1])
```

```
ADF statistics -6.383360838247469
Critical value @ 0.05  -2.8619717403241975
p-Value  2.194469400634231e-08
```

Figure 4 23 : Augmented Dickey Fuller test

### 4.8.2 Autocorrelation

Parameter Autocorrelation indicates how the movement of the past observations and future observation are related to each other. Past observations are considered as independent variable and future observation as dependent variable. In Arima model AR(p) term is the highest lag observation to include in our regressive model, that is how many past observations we should use to forecast the future observation. Figure 4 24 is the Autocorrelation graph for the temperature data set. We see that value of 1 has the maximum value and can be used as p parameter.

```
In [87]:
from statsmodels.graphics.tsaplots import plot_acf

plt.figure(figsize = (12, 4))
plot_acf(trainDf, ax = plt.gca(), lags = 30)
plt.show()
```

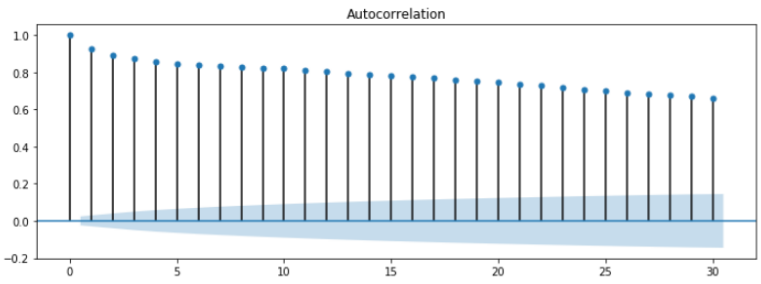



Figure 4 24 : Autocorrelation graph

### 4.8.3 Partial autocorrelation

Auto correlation which we discussed in 4.7.2 also captures the indirect relationships between the lags. However, the Arima model also needs the direct relationship between the lags. To get this we use the partial auto correlation which captures direct relationships between the

lags with the relationship between the intermediate lag observations removed. Figure 4 25 shows the Partial autocorrelation graph. From the graph we see that value is maximum at 1 thus we will be using parameter moving average parameter (maximum window size, MA) as 1.

```
In [88]:
from statsmodels.graphics.tsaplots import plot_pacf

plt.figure(figsize = (12, 4))
plot_pacf(trainDf, ax = plt.gca(), lags = 30)
plt.show()
```

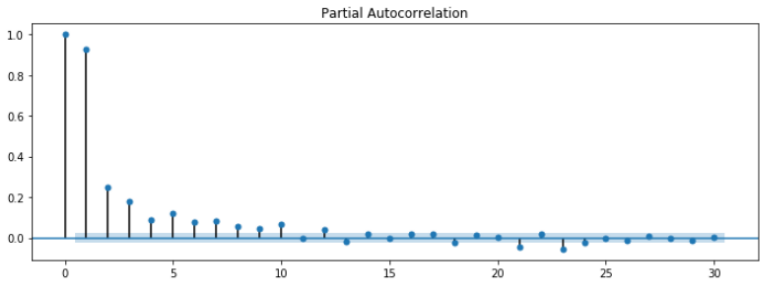



Figure 4 25 : Partial Autocorrelation graph

### 4.8.4 Modelling

We have created the model the model using p, q = 1 and d = 0 as analyzed in previous sections. Figure 4 26 shows the summary of the model. From the coefficients values it is clear that the p value for all the terms is significant. Constant coefficient is actually the mean value of the time series.

In [75]:

```
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(trainDf['_tempm'], order = (1, 0 , 1))
result = model.fit()

result.summary()
```

Out[75]:

ARMA Model Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | _tempm | **No. Observations:** | 6732 |
| **Model:** | ARMA(1, 1) | **Log Likelihood** | -15738.945 |
| **Method:** | css-mle | **S.D. of innovations** | 2.506 |
| **Date:** | Thu, 27 May 2021 | **AIC** | 31485.891 |
| **Time:** | 11:27:01 | **BIC** | 31513.149 |
| **Sample:** | 11-01-1996 | **HQIC** | 31495.302 |
| | - 04-07-2015 | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 26.4087 | 0.757 | 34.893 | 0.000 | 24.925 | 27.892 |
| **ar.L1._tempm** | 0.9758 | 0.003 | 333.421 | 0.000 | 0.970 | 0.982 |
| **ma.L1._tempm** | -0.3961 | 0.016 | -24.888 | 0.000 | -0.427 | -0.365 |

Roots

| | Real | Imaginary | Modulus | Frequency |
|---|---|---|---|---|
| **AR.1** | 1.0248 | +0.0000j | 1.0248 | 0.0000 |
| **MA.1** | 2.5244 | +0.0000j | 2.5244 | 0.0000 |

Figure 4 26 : ARIMA Model Summary

Figure 4 27 shows the ARIMA equation, predicting the temperature using the previous observation and error value:

$$\hat{y} = \beta_0 + \beta_1 S_{t-1} + \phi_1 \varepsilon_{t-1}$$

Figure 4 27 : ARIMA Equation

Where value of constants is as below:

β0 = 26.41

β1 = 0.98

Ø1 = -0.4

## 4.9 Summary

In this section we started with understanding of our data set we saw that there were missing values present in various columns. We tuned a particular threshold value based upon which we decided to drop the irrelevant columns. Based upon the percentage of missing/nan values we also decided that mean is the best metrics which can be used for imputation. We saw that

we should do imputation for missing values less than 6 percentage, columns having greater than that were cleaned by removing the corresponding rows. In section 4.2.2 we performed the handling of categorical variables we checked the value count of different categorical variable and clubbed the various values based upon the frequency. This helped to reduce the dimension of these variables. Further we saw that there was presence of Outliers in the data set. We used box plots to analyze the outliers and performed quantile-based method to remove the data points which can cause problems during modelling procedure. In section 4.3 we performed EDA on the cleaned data set. We visualized various columns and saw that Delhi environment is smokey in nature and has very less rainfall. It also reveals high level of pollution and humidity. We also saw in detail the variation in temperature and that in month of May, June, July temperature reaches its peak. In section 4.4 we performed sampling on the data set, we converted the hourly temperature values to daily by taking the mean value of temperature across the day. In section 4.5 then we discussed the splitting of data into test and train. We performed the methods which we need in modelling for both Neural Network and ARIMA. Further in section 4.6 we did the time series data generation needed for Neural Network modelling, as well as created the input we needed to perform ARIMA modelling. Finally in section 4.6 we performed the Neural Network modelling we went through structure of the Neural network in detail. We saw the layers we used in the modelling and analyzed the input, output and trainable parameters for each layer. We saw main building block of our model that is one dimension convolution neural network and long-short term memory network. Other layers like dense, flatten were used for connectivity. Various hyper parameters were tuned and detailed in this section. This section also highlights the steps we took to avoid over fitting by deploying dropouts and batch normalization. In last section 4.7 we went through ARIMA model, we performed augmented dickey fuller test on the data set and found that our time series is already stationary and thus we did not need to use differencing. Finally, we plotted ACF and PACF graphs and found that value of AR(p) and MA(q) which best suits our data set are 1, 1.

# CHAPTER 5: RESULT AND DISCUSSION

## 5.1 Introduction

In this section we will check the finding of the modelling. We will see the output of the each of the models. We will then use means square error as our loss function and see the value of MSE for each model. We will further draw predictions on the test data set and try to visualize the output or prediction on the test data set.

## 5.2 Model Output

We run the Neural Network model we created in section 4.6. We have used 300 Epochs on the hyper parameter tunned in section 4.6. Figure 5 1 shows the output of first 10 and last 10 Epochs. From this and the Error graph, we can see that as Epoch's are increasing Error/loss is coming down nicely.

```
In [73]: 1 h = model.fit(x_data_train, y_data_train,epochs=300, verbose=1, callbacks = callbacks_list)
Epoch 1/300
6702/6702 [==============================] - 13s 2ms/sample - loss: 0.7237
Epoch 2/300
6702/6702 [==============================] - 11s 2ms/sample - loss: 0.3292
Epoch 3/300
6702/6702 [==============================] - 11s 2ms/sample - loss: 0.19190s -
Epoch 4/300
6702/6702 [==============================] - 12s 2ms/sample - loss: 0.1264
Epoch 5/300
6702/6702 [==============================] - 12s 2ms/sample - loss: 0.0848
Epoch 6/300
6702/6702 [==============================] - 13s 2ms/sample - loss: 0.0560
Epoch 7/300
6702/6702 [==============================] - 12s 2ms/sample - loss: 0.0437
Epoch 8/300
6702/6702 [==============================] - 12s 2ms/sample - loss: 0.03510s - l
Epoch 9/300
6702/6702 [==============================] - 14s 2ms/sample - loss: 0.0308
Epoch 10/300
6702/6702 [==============================] - 16s 2ms/sample - loss: 0.0272
```

```
Epoch 290/300
6688/6702 [============================>.] - ETA: 0s - loss: 0.0167
Epoch 00290: ReduceLROnPlateau reducing learning rate to 2.9385228796891414e-42.
6702/6702 [==============================] - 13s 2ms/sample - loss: 0.0167
Epoch 291/300
6702/6702 [==============================] - 13s 2ms/sample - loss: 0.01720s - lo
Epoch 292/300
6688/6702 [============================>.] - ETA: 0s - loss: 0.0170
Epoch 00292: ReduceLROnPlateau reducing learning rate to 1.4692614398445707e-42.
6702/6702 [==============================] - 13s 2ms/sample - loss: 0.0170
Epoch 293/300
6702/6702 [==============================] - 13s 2ms/sample - loss: 0.0173
Epoch 294/300
6688/6702 [============================>.] - ETA: 0s - loss: 0.0179
Epoch 00294: ReduceLROnPlateau reducing learning rate to 7.342803953062041e-43.
6702/6702 [==============================] - 14s 2ms/sample - loss: 0.0180
Epoch 295/300
6702/6702 [==============================] - 14s 2ms/sample - loss: 0.0175
Epoch 296/300
6688/6702 [============================>.] - ETA: 0s - loss: 0.0173
Epoch 00296: ReduceLROnPlateau reducing learning rate to 3.671401976531021e-43.
6702/6702 [==============================] - 14s 2ms/sample - loss: 0.0173
Epoch 297/300
6702/6702 [==============================] - 14s 2ms/sample - loss: 0.0174
Epoch 298/300
6688/6702 [============================>.] - ETA: 0s - loss: 0.0181
Epoch 00298: ReduceLROnPlateau reducing learning rate to 1.8357009882655104e-43.
6702/6702 [==============================] - 15s 2ms/sample - loss: 0.0181
Epoch 299/300
6702/6702 [==============================] - 15s 2ms/sample - loss: 0.0172
Epoch 300/300
6688/6702 [============================>.] - ETA: 0s - loss: 0.0174
```

```
In [73]: 1 pd.DataFrame(h.history)[['loss']].plot();
```

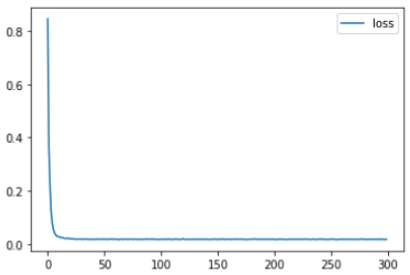
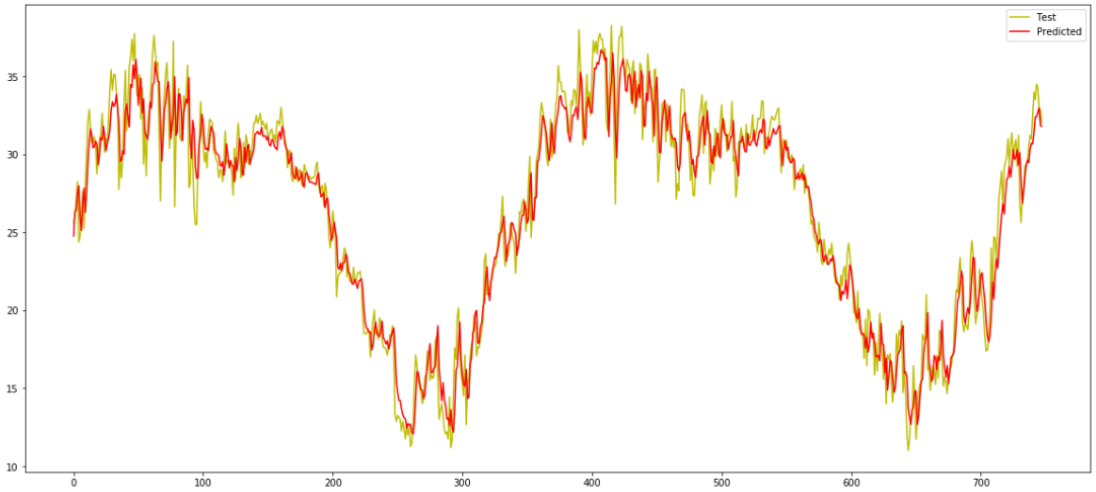



Figure 5 1 : Neural Network Execution

We than calculate the mean square error to check the accuracy of the model on test data set and draw the graph visualizing the prediction and the actual test output values. Figure 5 2 highlights red which shows the predicted temperature values while yellow shows the actual temperature value. This figure also shows that on the test data set we have achieved MSE of 2.59 which is quite good, and the visualization shows good prediction.

```
In [69]:
predict = model.predict(x_data_test)
predict = scalar.inverse_transform(predict)
Ytest = scalar.inverse_transform(y_data_test)
```

```
In [74]:
plt.figure(figsize=(20,9))
plt.plot(Ytest , 'y')
plt.plot(predict,'r')
plt.legend(('Test','Predicted'))
plt.show()
```




```
In [71]:
from sklearn.metrics import mean_squared_error
mean_squared_error(Ytest, predict)

Out[71]: 2.5891236446962242
```

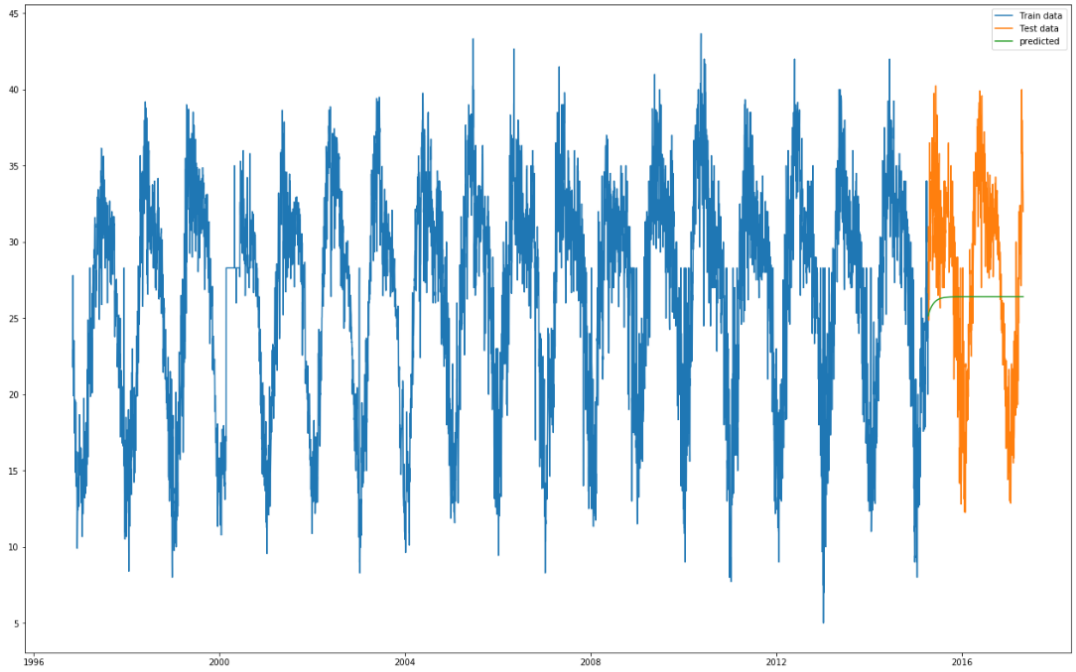
Figure 5 2 : Neural Network visualization and MSE

Now we run the prediction on the ARIMA model which was created in section 4.7. We see from Figure 5 3 that model is kind of just predicting the mean which is highlighted by the green color and is not able to follow the trend which is seen in the temperature data set (shown by the yellow for the test data set). We also see that the mean square error achieved by the ARIMA model is around 43.17 which is not good. Comparatively our Neural network model has performed far better.

```
In [177]:
tempDataDf['tempm_pred'] = result.predict(tempDataDf.index.min(), tempDataDf.index.max())
```

```
In [178]:
plt.figure(figsize = (22, 14))

plt.plot (trainDf['_tempm'], label = 'Train data')
plt.plot (testDf['_tempm'], label = 'Test data')
plt.plot (tempDataDf['tempm_pred'][testDf.index.min() : ], label = 'predicted')
plt.legend(loc = 'best')
plt.show()
```




```
In [179]:
from sklearn.metrics import mean_squared_error
mean_squared_error(testDf['_tempm'], tempDataDf['tempm_pred'][testDf.index.min() : ])
```

```
Out[179]: 43.17118457744932
```

Figure 5 3 : Arima Model visualization and MSE

## 5.3 Summary

We saw in this section the output and prediction accuracy for both Neural Network and ARIMA model. It was clearly seen that Neural Network performed good on the tunned hyper parameters with mean square error of 2.58 while ARIMA model does not seem to capture the trend of the temperature variation that well and was able to predict the mean temperature on the test data set. It is achieving mean square error of 43.17. There can be future enhancements which can be made on the ARIMA model which can further decrease the mean square error and provide good prediction results. We will discuss these points in the conclusion section which will bring forward further scope of work and future recommendations.

# CHAPTER 6: CONCLUSION AND FUTURE SCOPE

## 6.1 Introduction

In this section we will draw conclusion on the results we achieved for Neural Network and ARIMA modelling. We will draw conclusion on the performance of Neural Network and ARIMA modelling on our weather data set. We will discuss the insight the output of the modelling provides on the time series prediction. Lastly we will close with the future recommendation and further study we can do on this topic. We will bring forward the short coming we noticed in this study and what future implementation can be analyzed to achieve better results on time series forecast of daily temperature.

## 6.2 Conclusion and Future Recommendation

After our modelling and prediction of test data set we saw that Neural Network build from one dimensional Convolution Neural Network, long short-term memory networks and other components performs quite well on timeseries weather data we are using. Mean square error achieved was below 3, thus we can conclude that these networks can be used for predicting time series data sets like weather data. We also noticed that we did not had too deep Neural Network and relatively model (with around 3 lakh tunable parameters) was able to train on nominal resources with good prediction output. From ARIMA modelling we saw that we did not achieved good mean square error and our model was precisely just able to predict the mean temperature value while the variation in temperature was not captured.

In the future scope of this study, we can analyze the seasonal decomposition of the weather data and look for seasonality and trends based upon that it is recommended to use models like SARIMA ( Seasonal Auto Regressive Moving Average) which includes seasonal components during training of the model.

**REFERENCES**


Garima Jain, B. M., 2016. *A Review on Weather Forecasting Techniques.* Vol. 5, Issue 12 ed. Uttar Pradesh, India: International Journal of Advanced Research in Computer and Communication Engineering.

Gopinath N, V. S. P. P. J. A. D. S., 2020. *Weather Prediction using Machine Learning and IOT.* Volume-9 Issue-4 ed. s.l.:International Journal of Engineering and Advanced Technology (IJEAT).

Jennifer L. Cardona, M. F. H. J. O. D., 2019. *Seeing the Wind:VisualWind Speed Prediction with a Coupled Convolutional and Recurrent Neural Network.* arXiv:1905.13290 [stat.ML] ed. Stanford: arXiv:1905.13290 [stat.ML].

Jonathan A. Weyn, D. R. D. R. C., 2020. *IMPROVING DATA-DRIVEN GLOBAL WEATHER PREDICTION USING DEEP CONVOLUTIONAL NEURAL NETWORKS ON A CUBED SPHERE.* arXiv:2003.11927v1 ed. seattle: arXiv.

Kumar, R., 2013. *Decision Tree for the Weather Forecasting.* Volume 76 ed. Gurgaon, India: International Journal of Computer Applications.

N Anusha, M. S. C. G. J. R., 2019. *Weather Prediction Using Multi Linear Regression Algorithm.* volume 590 ed. Chennai, India : International Conference on Frontiers in Materials and Smart System Technologies.

P.Kalaiyarasi, M., 2018. *Data Mining Techniques Using To Weather Prediction.* Volume 6 Issue 3 ed. Arachaur, India: International Journal of Computer Science Trends and Technology (IJCST).

Pabreja, K., 2012. *Clustering technique for Interpretation of Cloudburst over leh.* Vol. 1 • No. 2, 2012 ed. pilani, india: CSI Journal of Computing.

Priyanka Mahajan, C. N. S. K. P. K. S., 2017. *Weather Forecasting using Neural Network.* ICIATE - 2017 ed. Mumbai, India: International Journal of Engineering Research & Technology (IJERT).

S.Veenadhari, B. M. C. S., 2014. *Machine learning approach for forecasting crop yield based on climatic parameters.* Jan. 03 –05 (ICCCI -2014), ed. coimbtore, indi: International Conference on Computer Communication and Informatics.

Sarah Chapman, .. J. E. M. W. A. S. M. T. C. A. M., 2017. *The impact of urbanization and climate change on urban temperatures.* 10.1007/s10980-017-0561-4 ed. s.l.:Springer.

ZhanJie Wang, A. B. M. M. M., 2017. *The Weather Forecast Using Data Mining Research Based on Cloud Computing..* doi :10.1088/1742-6596/910/1/012020 ed. China: IOP Conf. Series: Journal of Physics: Conf. Series 910 (2017).

Anjali, T., Chandini, K., Anoop, K. and Lajish, V.L., (2019) Temperature Prediction using Machine Learning Approaches. *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies, ICICICT 2019*, pp.1264–1268.

Bangaru Kamatchi, S. and Parvathi, R., (2020) Crop- Climate management: A review. *Journal of Critical Reviews*, 76, pp.871–880.

Biradar, P., Ansari, S., Paradkar, Y. and Lohiya, S., (2017) Weather Prediction Using Data Mining. *Internatio nal Journal of Engineering Development and Research*, [online] 2, p.211. Available at: www.ijedr.org.

Cifuentes, J., Marulanda, G., Bello, A. and Reneses, J., (2020) Air temperature forecasting using machine learning techniques: A review. *Energies*, 136, pp.1–28.

Corresponding, S.S. De, (2009) Artificial Neural Network Based Prediction of Maximum and Minimum Temperature in the Summer Monsoon Months over India. pp.37–44.

Dhore, A., Byakude, A., Sonar, B. and Waste, M., (2017) Weather Prediction using the Data Mining Techniques. *International Research Journal of Engineering and Technology*, 45, pp.2562–2565.

Ganju, A., (2012) Maximum and minimum temperature prediction over western Himalaya using artificial neural network. 2April, pp.283–290.
Journal, I. and Technology, A., (2020) Weather Prediction using Machine Learning and IOT. *International Journal of Engineering and Advanced Technology*, 94, pp.2094–2098.

Kapoor, P. and Bedi, S.S., (2013) Weather Forecasting Using Sliding Window Algorithm. *ISRN Signal Processing*, 2013, pp.1–5.

Liao, S.H., Chu, P.H. and Hsiao, P.Y., (2012) Data mining techniques and applications - A decade review from 2000 to 2011. *Expert Systems with Applications*, [online] 3912, pp.11303–11311. Available at: http://dx.doi.org/10.1016/j.eswa.2012.02.063.

Maiti and Bidinger, (1981) Performance Metrics in machine learning. *Journal of Chemical Information and Modeling*, 539, pp.1689–1699.

Majumdar, J., Naraseeyappa, S. and Ankalaki, S., (2017) Analysis of agriculture data using data mining techniques: application of big data. *Journal of Big Data*, 41.

Mohammed, M., Kolapalli, R., Golla, N. and Maturi, S.S., (2020) Prediction of rainfall using machine learning techniques. *International Journal of Scientific and Technology Research*, 91, pp.3236–3240.

Muqeem, M. and Javed, N., (2016) A Critical Review of Data Mining Techniques in Weather Forecasting. 54, pp.1091–1094.

Periyasamy, A.R.P. and Devi, P.S., (2017) Density Based Clustering Technique on Crop Yield Prediction. *International Journal of Advanced Research in Computer Science and Software Engineering*, 74, pp.246–250.

Ridwan, W.M., Sapitang, M., Aziz, A., Kushiar, K.F., Ahmed, A.N. and El-Shafie, A., (2020) Rainfall forecasting model using machine learning methods: Case study Terengganu, Malaysia. *Ain Shams Engineering Journal*, [online] xxxx. Available at: https://doi.org/10.1016/j.asej.2020.09.011.

Shah, U., Garg, S., Sisodiya, N., Dube, N. and Sharma, S., (2018) Rainfall Prediction : Accuracy Enhancement Using Machine Learning and Forecasting. pp.20–22.

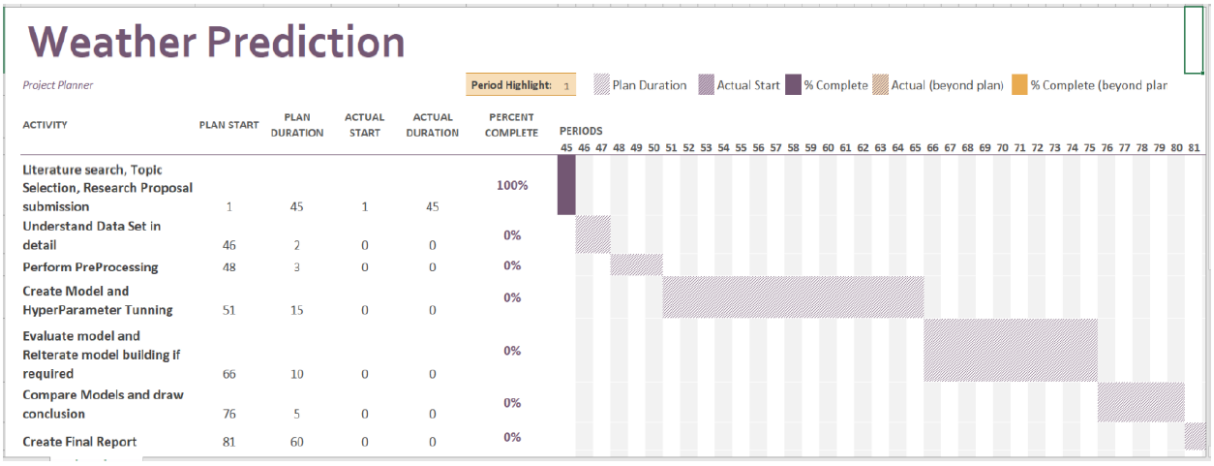
Shinde, P.P., Oza, K.S. and Kamat, R.K., (2017) Big data predictive analysis: Using R - analytical tool. *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, pp.839–842.

Souza, R.C., (2000) Combining Neural Networks and ARIMA Models for Hourly Temperature Forecast ( 1 ) Federal University of Juiz de Fora , Statistical Dept ., Brazil ( on leave at London Business School , U . K .). 12, pp.414–419.

Sutar, K., (2020) Low Cost Wireless Weather Monitoring System. *International Journal of Engineering Technologies and Management Research*, [online] 11, pp.35–39. Available at: http://dx.doi.org/10.29121/ijetmr.v1.i1.2015.24.

Tadayon, M. and Iwashita, Y., (n.d.) Comprehensive Analysis of Time Series Forecasting Using Neural Networks. pp.1–9.

## APPENDIX A: RESEARCH PLAN



## APPENDIX B: RESEARCH PROPOSAL

# PREDICTING WEATHER INFORMATION USING MACHINE LEARNING METHODS AND ARTIFICIAL INTELLIGENCE

RAVNEET SINGH, MSC MACHINE LEARNING

RESEARCH PROPOSAL

JANUARY 2021

**Abstract**

Weather has been playing an important role in day-to-day life activities. The impact of weather is enormous on our basic needs. Various fields highly depend upon the weather conditions and requires this to work in appropriation. Given the gathered data with various weather parameters such as temperature, humidity, dew levels etc. machine learning algorithms can be used to determine or predict the weather condition. This prediction can be used in many industries to better frame and plan their execution and take required steps to deal with abnormal conditions.

We will see and understand ML models which can be used to do this prediction. We will analyze Delhi Weather Data and apply Models like Decision trees, Random forests, Artificial Neural network. We will further explore the model architecture and performance shown by each model.

## 1. Background and Related Research

With increase in urbanization and other human intervention in nature, there have been vast change in weather and environmental conditions. Below paper provides a good summary of how urbanization is affecting the temperature conditions (Sarah Chapman, 2017).

Predicting of weather condition has applicability in wide variety of fields. Various natural phenomenon like cyclones, rainfall, draughts, sea conditions etc. has wide range of effects on daily life. Predicting the weather condition with high precision is important to avoid huge loss to the various dependent fields like crops and agriculture, irrigation department, forest department, travel domain, environmental science etc. In our day-to-day activity having a prediction of weather condition is highly helpful to plan our tasks with efficiency and better output. It can also help to take care life of the living beings on this planet better and can convey information to others who are in danger due to extreme weather conditions. Due to recent changes in climate/weather condition this prediction gives us pointers for taking preventive measure to restore our climate and slow its deterioration. It is very important to train the ML algorithm with high quality data set which has been thoroughly cleaned and preprocessed. This should ensure that quality of prediction is good on unseen data. Weather data can be retrieved from vast variety of sources such has climate departments, satellite data sets etc. Agriculture is one of the main departments which heavily depends on climate/weather prediction. Appropriate steps can be taken for marketing, storage and

harvesting of the crops based upon these predications. In Paper (S.Veenadhari, 2014) a software tool named 'Crop Advisor' is created to indicate influence of weather conditions on Crop yield. Here various climate/weather parameters are considered and individual effect on the Crop yield is predicted. Predication accuracy of developed model on soyabean crop has been provided. Another impacting area is the cloud burst, which can lead to excessive amount of rainfall and can have destructive effect on various industries.

Paper (Pabreja, 2012) develops models for prediction of cloudbursts using clustering techniques. It works on the moisture distribution, vertical motion, sheer and creates a model which can do prediction of 16 days in advance. Having information of Storms in particular area is also one of the major applications of weather prediction. Accurate prediction of stormy condition prior in time can help us take preventive steps which proves to be very helpful to save lives and related damages.

Wind related parameters proves to be really helpful in such predictions. Live sensors are often used to measure these parameters on real time which are used as input to the trained model to get the forecasting results. Convolution Neural Network is also often used as a feature extractor, output of which is fed into a Recurring neural network. Paper (Jennifer L. Cardona, 2019) explores the video capture to predict the same. This paper also brings forward the usage of LSTM (long short-term memory) networks for prediction of turbulent wind speeds.

Paper (Jonathan A. Weyn, 2020)works on weather prediction using Deep Convolution Neural Network on a cubed sphere to predict various atmospheric variables. Though this model is less accurate than other forecasting models, but it provides a good comparative study of implementation on given data set. There are many studies which are done in weather prediction on the previous years-maintained data. Along with various data mining and other traditional techniques, with growth of Machine learning and Artificial Intelligence there has been considerable increase in prediction accuracy and methods used for this purpose. Below paper explores data mining techniques for weather prediction (P.Kalaiyarasi, 2018). It researches on available methods for weather prediction and goes through the data mining stages. There are researches which work on predicting weather condition and predicting rainfall, this research uses real time data through sensors and use the data set to get the prediction on rainfall levels (Gopinath N, 2020). There are also researches like (ZhanJie Wang, 2017) on Weather forecast and Cloud computing. Cloud computing adds the advantages of using the efficient, secure, reliable data storage and therefore reduces cost and infrastructure required for weather data storage. Paper (Garima Jain, 2016) explores the time

series and correlation models to understand the weather forecast. Time series analysis help to understand and build the stochastic nature of the model from given observational series. Further it helps us to forecast or predict values on the unseen data from the previous data observations. Decision trees have been used to predict the weather forecast, a research (Kumar, 2013) explores the usage of decision trees and brings forward various techniques like Gini Index and information gain for the prediction of weather information. Research is also being made on Regressive models to predict the weather conditions, paper (N Anusha, 2019) executes a Regression model to predict the rainfall conditions. With advance of the Machine learning algorithms there have been research on the Artificial Neural networks, below paper bring forward this approach (Priyanka Mahajan, 2017). Approach to handle weather prediction using Artificial Neural Network is discussed. Since weather data is generally nonlinear and require a complex model to formulate the prediction on the data set, Neural Network is often a good choice.

## 2. Problem Statement

We are working on problem of predicting weather information on Delhi, India data set using machine learning algorithms and Artificial intelligence.
Machine learning algorithms will be used namely Decision Trees, Random Forest, Artificial Neural network to predict the weather conditions.
Based on the result metrics we will bring forward why and which model gives good prediction in terms of time and accuracy. We will also notice tuned hyper parameters and model structure which can help us to provide accurate results on unseen data.

## 3. Aim and Objectives

The main aim of this research is to propose machine learning model- Decision trees, Random Forest, Neural Network for weather prediction on Delhi, India data set. Identify the weather condition on test data set and compare the prediction accuracy.
Performance will be evaluated after training the model on data set which is properly cleaned and pre-processed.
The research objectives are formulated based on the aim of this study which are as follows:

- To analyze the pattern and relationship between the various weather parameters such as temperature, humidity, wind etc.

- Check and apply suitable techniques for preprocessing to handle any data imbalance and missing values if present.
- Compare the performance metrics of the predictive models and analyze which model is best suited for the weather data used.

## 4. Significance of the Study

This study will use machine learning models Decision Trees, Random Forest, Artificial Neural Network to predicate the weather condition on Delhi, India data set.
The resulting model can be used on unseen test data to get the weather condition, this can be used to make decisive planning steps based on the output.
We will try to fine tune our model based upon the different hyperparameters of the predictive model.
The accuracy comparison will provide an evidential conclusion on the model which performs better for the weather forecast data set.
It will also help us to compare the structure of the model example number of layers and neurons used in Neural Network to get a balance on the complexity of the hyperparameters versus the performance output gained.

## 5. Scope of the Study

Scope of the study will include:
Analyzing the data set and perform required cleaning and preprocessing.
Formulate Decision Tree, Random Forest, and Artificial Neural Network models on the given data set.
Perform hyper parameter tuning so as to achieve maximum possible accuracy with given resources.
Perform comparison and analysis on the models used and highlight the accuracy achieved on given model structure.

## 6. Research Methodology

### 6.1. Introduction

We will be using Delhi, India data set and will apply machine learning models to predict the weather conditions.

Focus will be to check analyze the data set in detail perform cleaning and data pre-processing on the columns if required.

Modelling will be done on given data set to perform prediction on the test data.

The model structure and accuracy details will help to decide the behaviour of the current model on the predication of weather forecast data.

### 6.2. Data Set Description

Below is the data set which is used for the research.

https://www.kaggle.com/mahirkukreja/delhi-weather-data

The above data set has more than 1 lakh rows.

Below are the columns present in the data set:

dewPtm (dew measurements)
fog
hail
heatindex
hum (humidity)
pressurem (pressure measurements)
rain
snow
tempm
thunder
tornado
vism (vision levels)
wdird, wdire, wgustm, windchillm, wspdm (wind parameters)

### 6.2. Data Pre-processing

It is one of the most important steps which should be performed on the data set before performing any modelling.

We will perform pre-processing steps which will include:

1.) Trying some EDA on the data columns.
2.) Checking and removing outliers.
3.) Check for percentage of missing values.
4.) Perform missing value substitution if required.
5.) Check for categorical data columns and perform necessary conversion for inputting these into the model.
6.) Check for any feature scaling which needs to be applied.
7.) Use appropriate technique to perform splitting of data for modelling process.

### 6.3. Models

We will perform Decision Trees, Random Forest, Artificial Neural Network modelling on the given data set.

**Decision Trees:**

Is very intuitive model which help us to perform classification based on modelled set of prediction rules.

It does not require data to be normalized or scaled and can also handle missing values in the data set up to some level.

Overall, it requires less effort on data pre processing however it require large training time and can often go complex, to overcome this effect we will also try Random forest on our data set.

**Random Forest:**

Is based on bagging algorithm and is basically a collection/ensemble of decision trees, can be used more effectively for classification tasks.

Radom forest can help over come overfitting of data and is seen to be more stable.
On the other hand, it can require more computation power and resources, taking longer training time.

**Artificial Neural Networks**:

These are one of the most popular Machine Learning algorithms which are very widely used. They have found application in many domains and if trained efficiently are quite robust.
This Model is based on human brain they can be used effectively on time series data as well.

## 7. Expected Outcomes

Once all the model creation is done, we need do prediction on the test data set to see the performance of the various models.
Performance/Accuracy of the models can be checked using various Evaluation metrices used in machine learning. We plan to use one or many of these for evaluation:

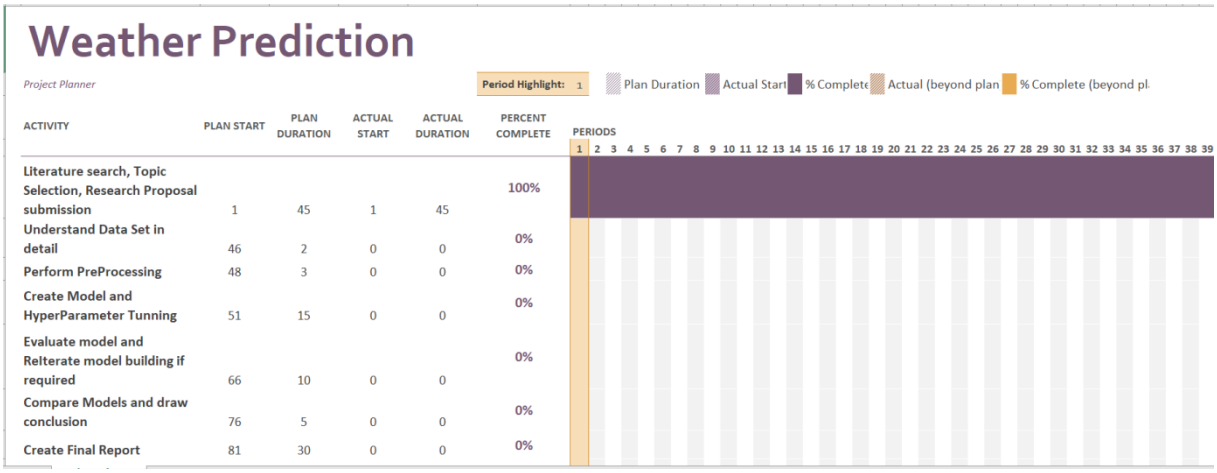
1.) **Confusion Matrix**: Help to analyse the True positive, True Negative, False positive, False Negative cases predicted by the model. Various metrics like sensitivity, specificity, accuracy can be derived to see performance of the model.
2.) **ROC curve** (Receiver operating characteristic curve): helps to see the True positive rate and false positive rate for a classification model.
3.) **AUC** (Area under the ROC curve): helps to measure the prediction quality of the model.

## 8. Requirements / resources

As we will be implementing the machine learning algorithms on python.
We would require a laptop or a machine on which Anaconda can be installed.
Using anaconda, we can easily use Jupiter notebook (python) to create machine learning models using various python API available for same.

We will also require GPU access to train our Artificial Neural Network and perform hyper parameter tuning on it.

## 9. Research Plan




## References


Garima Jain, B. M., 2016. *A Review on Weather Forecasting Techniques.* Vol. 5, Issue 12 ed. Uttar Pradesh, India: International Journal of Advanced Research in Computer and Communication Engineering.

Gopinath N, V. S. P. P. J. A. D. S., 2020. *Weather Prediction using Machine Learning and IOT.* Volume-9 Issue-4 ed. s.l.:International Journal of Engineering and Advanced Technology (IJEAT).

Jennifer L. Cardona, M. F. H. J. O. D., 2019. *Seeing the Wind:VisualWind Speed Prediction with a Coupled Convolutional and Recurrent Neural Network.* arXiv:1905.13290 [stat.ML] ed. Stanford: arXiv:1905.13290 [stat.ML].

Jonathan A. Weyn, D. R. D. R. C., 2020. *IMPROVING DATA-DRIVEN GLOBAL WEATHER PREDICTION USING DEEP CONVOLUTIONAL NEURAL NETWORKS ON A CUBED SPHERE.* arXiv:2003.11927v1 ed. seattle: arXiv.

Kumar, R., 2013. *Decision Tree for the Weather Forecasting.* Volume 76 ed. Gurgaon, India: International Journal of Computer Applications.

N Anusha, M. S. C. G. J. R., 2019. *Weather Prediction Using Multi Linear Regression Algorithm.* volume 590 ed. Chennai, India : International Conference on Frontiers in Materials and Smart System Technologies.

P.Kalaiyarasi, M., 2018. *Data Mining Techniques Using To Weather Prediction.* Volume 6 Issue 3 ed. Arachaur, India: International Journal of Computer Science Trends and Technology (IJCST).

Pabreja, K., 2012. *Clustering technique for Interpretation of Cloudburst over leh.* Vol. 1 • No. 2, 2012 ed. pilani, india: CSI Journal of Computing.

Priyanka Mahajan, C. N. S. K. P. K. S., 2017. *Weather Forecasting using Neural Network.* ICIATE - 2017 ed. Mumbai, India: International Journal of Engineering Research & Technology (IJERT).

S.Veenadhari, B. M. C. S., 2014. *Machine learning approach for forecasting crop yield based on climatic parameters.* Jan. 03 –05 (ICCCI -2014), ed. coimbtore, indi: International Conference on Computer Communication and Informatics.

Sarah Chapman, .. J. E. M. W. A. S. M. T. C. A. M., 2017. *The impact of urbanization and climate change on urban temperatures.* 10.1007/s10980-017-0561-4 ed. s.l.:Springer.

ZhanJie Wang, A. B. M. M. M., 2017. *The Weather Forecast Using Data Mining Research Based on Cloud Computing..* doi :10.1088/1742-6596/910/1/012020 ed. China: IOP Conf. Series: Journal of Physics: Conf. Series 910 (2017).