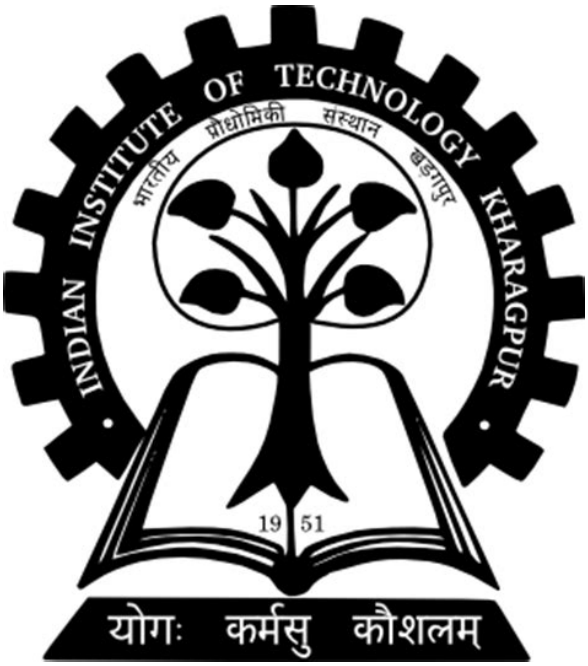
WEATHER FORECASTING OF KHARAGPUR

-TERM PROJECT, BIG DATA(MA60306)



DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

April 2023

Team Members :

Mohit Sharma (22MA60R20 , mohit22@kgpian.iitkgp.ac.in),
Dinesh(22MA60R05), Uday Sankar(22MA60R35), Vaibhav Kumar(22MA60R21), Joy Saha(22MA60R25), Pallab Pal(22MA60R16), Shikhar Shukla(22MA60R19), Himali(22MA60R12), Soumya Ganai(22MA60R07), Shashi Tarode(22MA60R23).

CONTENTS

1. PROBLEM FORMULATION

Weather forecasting uses science and technology to forecast atmospheric conditions for a particular place and period. Predicting weather can be helpful in industries, agriculture, tourism, construction, sports, transportation, disaster, and energy management. It is also essential to accurately predict the weather to ensure that everyday activities are straightforward and without difficulty, especially in light of the fact that the entire world is experiencing ongoing environmental change and its effects.

Over the years, several approaches have been developed to make weather forecasting more accurate and efficient. One such approach is using machine learning algorithms, which have shown great promise in predicting weather patterns. These algorithms are trained on historical weather data, such as temperature, humidity, wind speed, and pressure, and use that data to make predictions about future weather conditions. This process involves a feedback loop, where the accuracy of the predictions is evaluated against the actual weather conditions to improve the algorithm's performance.

Machine learning algorithms have been used in weather prediction for several years. However, recent advancements in computing power, data storage, and the availability of large amounts of data have led to significant improvements in weather forecasting accuracy. Machine learning models have been used to accurately predict rainfall, temperature, wind patterns, and storm systems, and they have become an essential tool for meteorologists and climatologists.

In this project, we utilize machine learning algorithms to predict the upcoming weather conditions in the Kharagpur area. The first chapter details the data collection, cleaning, and exploration processes. Chapters three and four outline the methodology and the results of our work, respectively.

2. DATA

2.1 Data Collection

Weather data is kept by numerous organisations, including the National Weather Service, Indian Space Research Organization, Indian Meteorological Department, and National Aeronautics and Space Administration (NASA). Storing weather data is important for various reasons other than Weather Forecasting. Weather data is also used to study long-term climate patterns, including trends in temperature, precipitation, and other meteorological variables. This information is used to develop models of how the climate is changing over time and to understand the impacts of climate change on ecosystems, human health, and infrastructure. Energy companies also require such data to forecast demand for energy, such as electricity or natural gas, based on factors like temperature, humidity, and wind speed. This information is used to manage energy supply and demand more efficiently, which can help to reduce costs and improve reliability.

The data for this project has been extracted from the NASA Power API. API stands for Application Programming Interface, which is a set of protocols, routines, and tools for building software applications.

	Unnamed: 0	PS	WS10M	WS10M_MAX	WS10M_MIN	WS10M_RANGE	WD10M	WS50M	WS50M_MAX	WS50M_MIN	...	PRECTOTCORR	ALLSKY_SFC_SW_DWN	CLRSKY_SFC_SW_DWN	ALLSKY_KT	ALLSKY_SFC_LW_DWN	ALLSKY_
0	20120101	101.14	1.80	2.78	0.65	2.13	124.50	3.07	5.93	0.63	...	1.31	150.58	176.69	0.53	380.00	
1	20120102	101.06	2.08	4.14	0.59	3.55	112.56	3.05	6.88	0.38	...	1.42	96.89	171.78	0.34	390.42	
2	20120103	101.11	2.16	2.59	1.75	0.84	153.69	3.32	4.46	1.94	...	0.46	123.61	161.46	0.44	385.35	
3	20120104	101.17	1.48	2.61	0.34	2.27	71.06	2.40	5.37	0.44	...	0.23	81.25	138.21	0.29	385.75	
4	20120105	101.21	1.50	2.38	0.76	1.62	127.88	2.25	4.55	1.01	...	0.27	98.29	123.83	0.34	379.76	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4103	20230327	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	...	-999.00	-999.00	-999.00	-999.00	-999.00	
4104	20230328	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	...	-999.00	-999.00	-999.00	-999.00	-999.00	
4105	20230329	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	...	-999.00	-999.00	-999.00	-999.00	-999.00	
4106	20230330	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	...	-999.00	-999.00	-999.00	-999.00	-999.00	
4107	20230331	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	-999.00	...	-999.00	-999.00	-999.00	-999.00	-999.00	

4108 rows × 31 columns

Fig. 2.1: Initial unprocessed data

APIs allow different software systems to communicate with each other and share data in a standardized way, making it easier to integrate different software components into a cohesive application. Data extracted from this API had 4108 records and 31 features including the date of observation. The 31 columns in the data are :

- T2M MERRA-2 Temperature at 2 Meters (C)
- T2MDEW MERRA-2 Dew/Frost Point at 2 Meters (C)
- T2MWET MERRA-2 Wet Bulb Temperature at 2 Meters (C)
- TS MERRA-2 Earth Skin Temperature (C)
- T2M_RANGE MERRA-2 Temperature at 2 Meters Range (C)
- T2M_MAX MERRA-2 Temperature at 2 Meters Maximum (C)
- T2M_MIN MERRA-2 Temperature at 2 Meters Minimum (C)
- QV2M MERRA-2 Specific Humidity at 2 Meters (g/kg)
- RH2M MERRA-2 Relative Humidity at 2 Meters
- PRECTOTCORR MERRA-2 Precipitation Corrected (mm/day)
- ALLSKY_SFC_SW_DWN CERES SYN1deg All Sky Surface Shortwave Downward Irradiance (kW-hr/m$\hat{2}$/day)
- CLRSKY_SFC_SW_DWN CERES SYN1deg Clear Sky Surface Shortwave Downward Irradiance (kW-hr/m$\hat{2}$/day)
- ALLSKY_KT CERES SYN1deg All Sky Insolation Clearness Index (dimensionless)
- ALLSKY_SFC_LW_DWN CERES SYN1deg All Sky Surface Longwave Downward Irradiance (W/m$\hat{2}$)
- ALLSKY_SFC_PAR_TOT CERES SYN1deg All Sky Surface PAR Total (W/m$\hat{2}$)
- CLRSKY_SFC_PAR_TOT CERES SYN1deg Clear Sky Surface PAR Total (W/m$\hat{2}$)
- ALLSKY_SFC_UVA CERES SYN1deg All Sky Surface UVA Irradiance (W/m$\hat{2}$)
- ALLSKY_SFC_UVB CERES SYN1deg All Sky Surface UVB Irradiance (W/m$\hat{2}$)
- ALLSKY_SFC_UV_INDEX CERES SYN1deg All Sky Surface UV Index
- WS2M MERRA-2 Wind Speed at 2 Meters (m/s)
- PS MERRA-2 Surface Pressure (kPa)
- WS10M MERRA-2 Wind Speed at 10 Meters (m/s)
- WS10M_MAX MERRA-2 Wind Speed at 10 Meters Maximum (m/s)
- WS10M_MIN MERRA-2 Wind Speed at 10 Meters Minimum (m/s)
- WS10M_RANGE MERRA-2 Wind Speed at 10 Meters Range (m/s)
- WD10M MERRA-2 Wind Direction at 10 Meters (Degrees)
- WS50M MERRA-2 Wind Speed at 50 Meters (m/s)

- WS50M_MAX MERRA-2 Wind Speed at 50 Meters Maximum (m/s)
- WS50M_MIN MERRA-2 Wind Speed at 50 Meters Minimum (m/s)
- WS50M_RANGE MERRA-2 Wind Speed at 50 Meters Range (m/s)
- WD50M MERRA-2 Wind Direction at 50 Meters (Degrees)

2.2 Data Preprocessing and Visualization

As evident from Figure 2.1, many NaN values in our data are represented as -999. We begin by converting all the -999 entries to NaN and converting the date feature to the date format YYYY-MM-DD. Next, we append some new features to our dataset such as "Weather type" and "Conditions." We will also train a classification model that predicts the weather type of the future date by first predicting the temperature and other variables for the same day.

Next, we analyze the correlation between the features and remove features that have a correlation more than 0.75. Among the two correlated features we try to remove the one that has more NaN values. In the resulting dataset we now have 14 features.

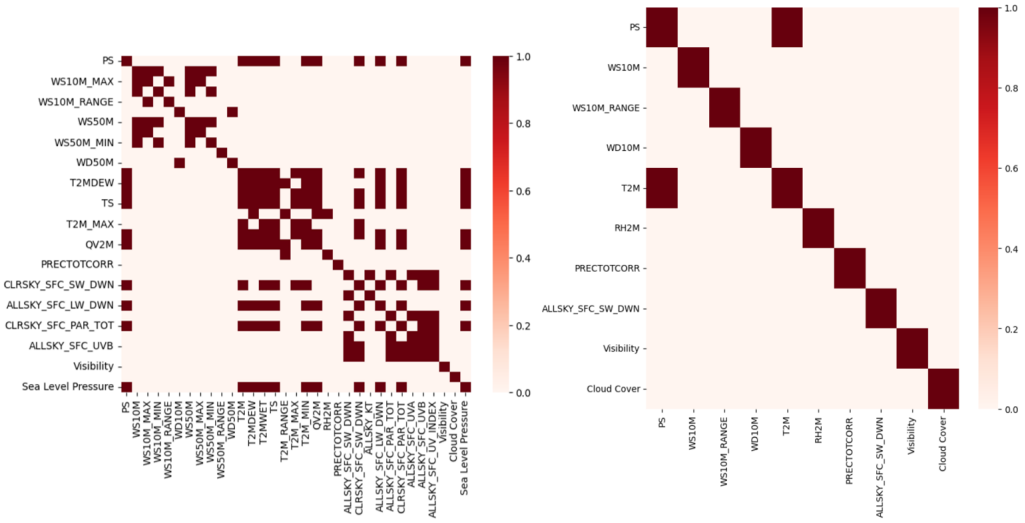


Fig. 2.2: Heatmap

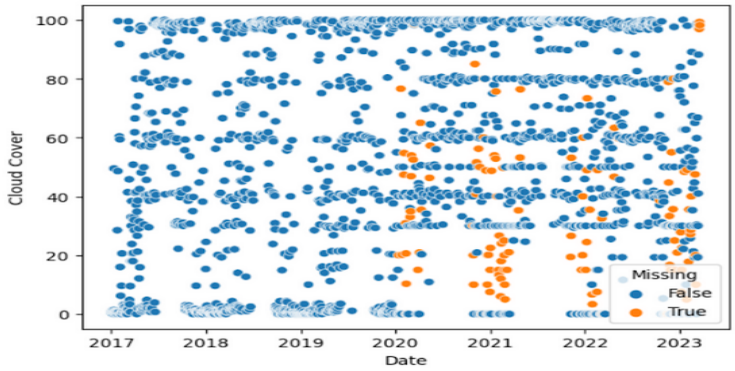


Fig. 2.3: Cloud Cover

Among the remaining features the feature "Cloud Cover" had the most NaN values. From the distribution of the feature, we noted that the variable is uniformly distributed. Therefore the interpolation strategy to fill the NaN values is the most plausible choice. It tends to work decently well with time series data because our dataset's first and last rows are not in the above slice of our rows. The interpolation method can vary, but we use a forward linear interpolation. To understand forward linear interpolation, suppose we have a column x with missing data points, and each row represents a day's entry-time series data. Let's say $x_0 = 0$ and that x_1 through x_4 are missing, but we have $x_5 = 5$. A forward linear interpolation makes a line between the points that we do have, so a straight line between $(0, 0)$ and $(5, 5)$ will fill in $x_t = t$ for $t = 1, 2, 3, 4$. Figure 2.3 shows the distribution of Cloud Cover along with the imputed missing values.

	Unnamed: 0	Date	PS	WS10M	WS10M_RANGE	WD10M	T2M	RH2M	PRECTOTCORR	ALLSKY_SFC_SW_DWN	Visibility	Cloud Cover	Conditions	weather_type
0	0	2017-01-01	101.27	0.86	1.29	142.25	19.87	72.81	0.00	138.71	2.5	0.4	Clear	Mist
1	1	2017-01-02	101.27	0.95	1.33	194.31	19.83	75.44	3.69	135.58	1.1	0.7	Clear	Fog
2	2	2017-01-03	101.24	2.49	2.16	325.44	18.69	83.31	2.25	134.17	4.0	1.0	Clear	Smoke Or Haze
3	3	2017-01-04	101.15	2.86	1.96	337.94	18.01	79.00	0.00	148.32	4.0	0.9	Clear	Smoke Or Haze
4	4	2017-01-05	101.13	2.80	2.20	333.69	17.16	72.50	0.00	166.91	4.0	0.7	Clear	Clear
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2263	2263	2023-03-16	100.76	3.98	5.09	174.88	26.80	64.75	2.49	199.41	3.9	97.6	Overcast	Overcast
2264	2264	2023-03-17	100.88	1.73	3.81	148.31	25.20	73.06	4.11	93.95	3.9	98.8	Overcast	Overcast
2265	2265	2023-03-18	100.77	1.32	3.14	215.88	26.46	65.06	2.20	93.95	4.0	96.9	Overcast	Overcast
2266	2266	2023-03-19	100.62	3.70	4.26	188.81	24.49	74.62	2.14	93.95	4.0	99.3	Overcast	Overcast
2267	2267	2023-03-20	100.48	3.66	2.95	207.44	25.83	61.31	1.34	93.95	3.9	98.2	Overcast	Overcast

2268 rows × 14 columns

Fig. 2.4: Processed Data

Now that we have our final data, let us see how some of the variables vary over time.

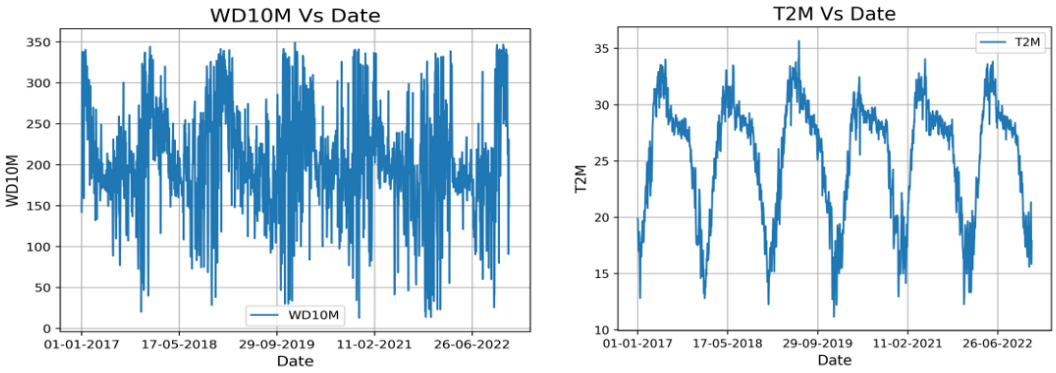


Fig. 2.5: Wind Direction and Temperature

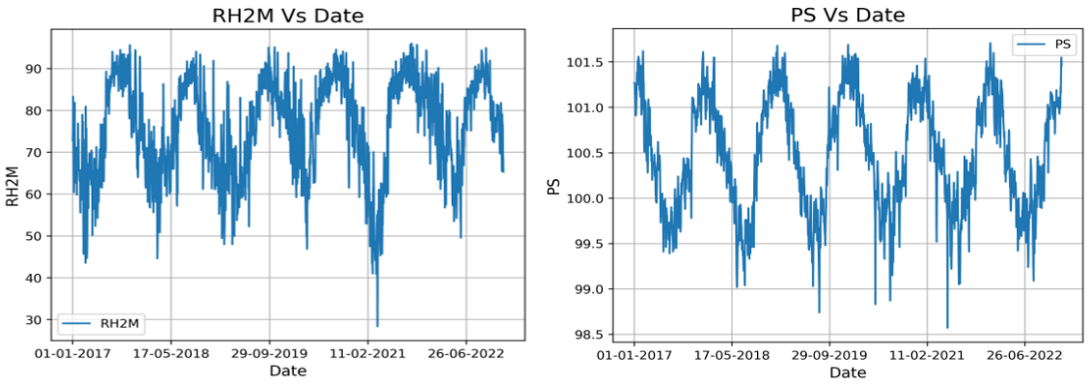


Fig. 2.6: Relative Humidity and Pressure

Autocorrelation graph represents the degree of similarity between a given time series and a lagged version of itself over successive time intervals i.e. it represents the relationship between a variable's current value and its past values. It helps us uncover hidden patterns in our data, helps us select the correct forecasting methods, and helps us identify seasonality in our time series data.

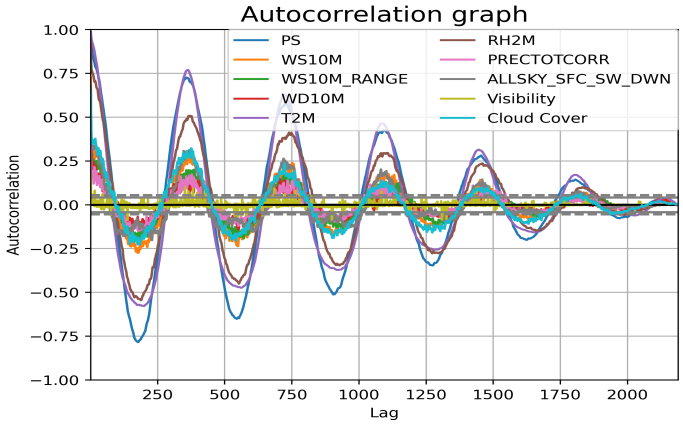


Fig. 2.7: Autocorrelation

3. METHODS

3.1 RECURRENT NEURAL NETWORKS

Consider a sequence $x = (x^{(1)}, x^{(2)}, ..., x^{(l)})$, where each element $x^{(i)} \in \mathbb{R}^d$ is a vector of dimension d. When training a traditional neural network on that data, we would feed in all information about this sequence in one go. This approach would ignore any temporal dependencies present in the sequence x. Furthermore, the number of weights in the network would increase linearly with the sequence length l.

Recurrent neural networks (RNNs) are designed to process sequential data more efficiently by taking into consideration the sequential nature of the data. A standard recurrent neural unit can process the sequence elements one at a time, starting with first element of the sequence before feeding in the second. At each time point, the unit takes two inputs: an element of the sequence $x_i \in \mathbb{R}^d$ and the output of the same unit at the previous time point, $h_{i-1} \in \mathbb{R}$. This allows the unit to process the whole sequence with a fixed number of weights, *i.e.*, the model is independent of the sequence length.

3.1.1 The structure of a single LSTM cell

Let $\mathbb{U} = [0, 1]$ represent the unit interval and let $\pm\mathbb{U} = [-1, 1]$. An LSTM cell has two recurrent features, denoted by h and c, called the hidden state and the cell state, respectively. The cell, denoted by $\mathcal{L}$, is a mathematical function that takes three inputs and produces two outputs:

$$(h(t), c(t)) = \mathcal{L}(h(t-1), c(t-1), x(t)),$$

where $h(t), h(t-1), c(t), c(t-1) \in \pm\mathbb{U}$ and $x(t) \in \mathbb{R}$. Both outputs leave the cell at time point t and are fed back into that same cell at time point $t+1$. At any time point t, an element of the input sequence $x(t) \in \mathbb{R}^d$ is also fed into the cell.

Inside the cell, the hidden state and the input vector are fed into three gates (functions), each of which produces a scalar value in $\mathbb{U}$ with the help of a sigmoid activation function:

$$f\_g^{(t)}(x^{(t)}, h^{(t-1)}) = \sigma(w_{f,x}^T x(t) + w_{f,h} h^{(t-1)} + b_f) \in \mathbb{U},$$

$$i\_g^{(t)}(x^{(t)}, h^{(t-1)}) = \sigma(w_{i,x}^T x^{(t)} + w_{i,h} h^{(t-1)} + b_i) \in \mathbb{U},$$

$$o\_g^{(t)}(x^{(t)}, h^{(t-1)}) = \sigma(w_{o,x}^T x^{(t)} + w_{o,h} h^{(t-1)} + b_o) \in \mathbb{U},$$

where $w_{f,x}, w_{i,x}, w_{o,x} \in \mathbb{R}^d and w_{f,h}, w_{i,h}, w_{o,h}, b_f, b_i, b_o \in \mathbb{R}$ are weight parameters (also called weight vectors and biases, respectively). These are the parameters to be learned during the training of the cell. The three gates can be interpreted as switches when their output values are near 1 (on) or 0 (off). Another scalar function, the so-called cell update (c_u), is constructed as a single neuron with a tanh activation function

$$\text{c\_u}^{(t)}(x^{(t)}, h^{(t-1)}) = \tanh(w_x^T x^{(t)} + w_h h^{(t-1)} + b) \in \pm\mathbb{U},$$

where $w_x \in \mathbb{R}^d$ and $w_{h,b} \in \mathbb{R}$ are further weight parameters to be learned. The forget gate (f_g) controls how much of the current cell state we should forget, the input gate (i_g) controls how much of the cell update is added to the cell state, and the output gate (o_g) controls how much of the modified cell state should leave the cell and become the next hidden state. Written in terms of mathematical functions, the new cell and hidden states at time t are

$$c^{(t)} = f\_g^{(t)} \cdot c^{(t-1)} + i\_g^{(t)} \cdot c\_u^{(t)} \in \mathbb{U},$$

$$h(t) = o\_g^{(t)} \cdot tanh(c^{(t)}) \in \pm\mathbb{U},$$

where the arguments $(x^{(t)}, h^{(t-1)})$ have been omitted readability. All of these functions and parameters are encapsulated in the function $\mathcal{L}$.

By its design using hidden states that pass through time, recurrent neural networks have the capability to take an input sequence of any length and produce an output sequence of any length.The user can decide at what time points to feed in the input sequence and at what time points to extract the outputs.

3.1.2 A layer of LSTM cells

A layer of n LSTM cells, which we denote by $\mathcal{L}_n$, corresponds to the concatenation of n cells $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_n$, each with a different set of internal weight parameters. That is,

$$
\begin{aligned}
(h_1^{(t)}, c_1^{(t)}) &= \mathcal{L}_1(h_1^{(t-1)}, c_1^{(t-1)}, \mathbf{x}^{(t)}),\\
(h_2^{(t)}, c_2^{(t)}) &= \mathcal{L}_2(h_2^{(t-1)}, c_2^{(t-1)}, \mathbf{x}^{(t)}),\\
&\vdots\\
(h_n^{(t)}, c_n^{(t)}) &= \mathcal{L}_n(h_n^{(t-1)}, c_n^{(t-1)}, \mathbf{x}^{(t)}),
\end{aligned}
$$

which can equivalently be written as

$$
(\mathbf{h}^{(t)}, \mathbf{c}^{(t)}) = \mathcal{L}_n(\mathbf{h}^{(t)}, \mathbf{c}^{(t)}, \mathbf{x}^{(t)}),
$$

where $\mathbf{h}^{(t)}, \mathbf{h}^{(t-1)}, \mathbf{c}^{(t)}, \mathbf{c}^{(t-1)} \in \pm\mathbb{U}^n$ and $\mathbf{x}^{(t)} \in \mathbb{R}^d$. The individual weight vectors and biases from each of the LSTMs can be stacked into matrices. The dot products become matrix-vector products and the scalar multiplications become element-wise multiplications. The activation functions are applied element-wise, allowing the simultaneous evaluation of a whole layer of LSTM cells. The three gates and the cell update function now contain weight matrices $W_{fx}, W_{ix}, W_{ox}, W_x \in \mathbb{R}^{n\times d}$ of size compatible with the input vector $\mathbf{x}^{(t)} \in \mathbb{R}^d$. The stacked hidden dimension is of dimension n and so the gates contain compatible weight matrices $W_{fh}, W_{ih}, W_{oh}, W_h \in \mathbb{R}^{n\times n}$ and bias vectors $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{b} \in \mathbb{R}^n$.

3.1.3 A multi-layer LSTM network

So far we have only considered a single layer of n LSTM cells, called $\mathcal{L}_n$. In practice, one often stacks multiple layers to increase the complexity of the function represented by the network. At each time point t, the function $\mathcal{L}_n$ has two outputs $\mathbf{h}^{(t)}, \mathbf{c}^{(t)} \in \pm\mathbb{U}_n$. The hidden states $h^{(t)}$ can be sequentially, into the next layer.

A multi-layer LSTM network can be thought of a function $\mathcal{S}$, where

$$
\begin{aligned}
&\vdots\\
(\mathbf{h}_{n_3}^{(t)}, \mathbf{c}_{n_3}^{(t)}) &= \mathcal{L}_{n_3}(\mathbf{h}_{n_3}^{(t-1)}, \mathbf{c}_{n_3}^{(t-1)}, \mathbf{h}_{n_2}^{(t)}),\\
(\mathbf{h}_{n_2}^{(t)}, \mathbf{c}_{n_2}^{(t)}) &= \mathcal{L}_{n_2}(\mathbf{h}_{n_2}^{(t-1)}, \mathbf{c}_{n_2}^{(t-1)}, \mathbf{h}_{n_1}^{(t)}),\\
(\mathbf{h}_{n_1}^{(t)}, \mathbf{c}_{n_1}^{(t)}) &= \mathcal{L}_{n_1}(\mathbf{h}_{n_1}^{(t-1)}, \mathbf{c}_{n_1}^{(t-1)}, \mathbf{x}^{(t)}),
\end{aligned}
$$

can be represented by

$$(\mathbf{H}^{(t)}, \mathbf{C}^{(t)}) = \mathcal{S}(\mathbf{H}^{(t-1)}, \mathbf{C}^{(t-1)}, \mathbf{x}^{(t)}).$$

Each layer in the network can have a different number of cells $n1, n2, n3, \ldots$, and so the hidden state and cell state vectors may be of different dimensions. The variables $\mathbf{H}^{(t)}$ and $\mathbf{C}^{(t)}$ represent the collection of all hidden states and cell states, respectively, at time point t. We remark that in many applications, a network of stacked LSTM cells might just be a building block for a much larger model. For example, in time series forecasting, an additional final layer is used to map the output from $\pm\mathbb{U}^n$, where n is the number of cells in the top layer, to time series values in $\mathbb{R}$.

3.2 ARIMA

ARIMA stands for Autoregressive Integrated Moving Average. It is a statistical model used for time series analysis and forecasting. ARIMA models are a class of linear models that take into account past values of a time series and the errors or residuals of past forecasts.

ARIMA models have three main components:

Auto - Regression (AR) - This refers to a regression model that uses lagged values of the dependent variable to predict future values. In other words, the current value of the time series is modeled as a linear combination of its past values. Number of AR terms (p): AR terms are just lags of dependent variable. For instance if p is 5, the predictors for $x(t)$ will be $x(t-1)$ .... $x(t-5)$.

Integration (I) - This refers to the process of differencing the time series to make it stationary. A stationary time series is one where the statistical properties such as mean, variance and autocorrelation are constant over time.

Moving Average (MA) - This refers to a model that uses past forecast errors or residuals to predict future values. In other words, the current value of the time series is modeled as a linear combination of past errors. Number of MA terms (q): MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for $x(t)$ will be $e(t-1)$ .... $e(t-5)$ where $e(i)$ is the difference between the moving average at ith instant and actual value.

3.2.1 Stationarity of data

In ARIMA modeling, it is important to ensure that the time series data is stationary before fitting the model. Stationarity is a statistical property of time series data where the statistical properties such as mean, variance, and autocorrelation are constant over time. If a time series is stationary, it is easier to model and make predictions because the statistical properties of the data do not change over time.

There are two types of stationarity: strict stationarity and weak stationarity. Strict

stationarity requires that the joint distribution of any set of values in the time series is the same for all time intervals. Weak stationarity requires that the mean, variance, and autocorrelation of the time series are constant over time. In practice, weak stationarity is more commonly used for modeling.

One way to test for stationarity is to use a unit root test such as the Augmented Dickey-Fuller (ADF) test. The ADF test tests the null hypothesis that a unit root is present in a time series, which indicates non-stationarity. If the null hypothesis is rejected, then the time series is stationary.

Another way to test for stationarity is to look at the time series plot and the autocorrelation function (ACF) plot. If the time series plot shows a trend or a systematic pattern, then the time series is likely non-stationary. If the ACF plot shows a slow decay in the autocorrelation coefficients over time, then the time series is likely non-stationary.

If the time series is found to be non-stationary, it can be made stationary through differencing. Differencing involves taking the difference between consecutive observations in the time series. If the differenced series is stationary, then an ARIMA model can be fitted to the differenced data. The order of differencing required to make a time series stationary depends on the behavior of the data and can be determined by examining the ACF and PACF plots of the differenced series.

3.2.2 PACF and ACF

PACF and ACF are two commonly used tools in time series analysis to understand the correlation structure of a time series.

ACF stands for Autocorrelation Function. It measures the correlation between a time series and a lagged version of itself. Specifically, the ACF at lag k is the correlation between the time series at time t and the same series lagged by k periods. ACF is used to identify the presence of autocorrelation in a time series.

PACF stands for Partial Autocorrelation Function. It measures the correlation between a time series and a lagged version of itself, controlling for the effects of all shorter lags. Specifically, the PACF at lag k is the correlation between the time series at time t and the same series lagged by k periods, after removing the effects of all shorter lags. PACF is used to identify the order of an ARIMA model.

Both ACF and PACF are useful tools for selecting appropriate models for a time series. By looking at the patterns in the ACF and PACF plots, we can identify the appropriate order of ARIMA models. For example, if the ACF plot shows a significant correlation at lag 1 and no significant correlations at other lags, then an AR(1) model may be appropriate. If the PACF plot shows a significant correlation at lag 1 and no significant correlations at other lags, then an MA(1) model may be appropriate.

3.2.3 Seasionality and SARIMA

Seasonality in time series data refers to the presence of patterns or fluctuations in the data that occur at fixed, regular intervals over time. These patterns can be related to the time of day, day of the week, month of the year, or any other regular time interval.

Seasonality can be modeled using a seasonal ARIMA (SARIMA) model. A SARIMA model is an extension of the ARIMA model that includes additional parameters to account for seasonality in the data. The order of a SARIMA model is denoted as $(p, d, q)(P, D, Q)s$, where p, d, and q are the same as in the ARIMA model, P, D, and Q are the seasonal orders, and s is the number of time periods in each season.

To identify the seasonality in the data, a seasonal decomposition analysis can be performed. This involves decomposing the time series data into three components: trend, seasonality, and residual. The trend component represents the long-term direction of the data, the seasonal component represents the repeating patterns in the data, and the residual component represents the random variation that cannot be explained by the trend or seasonality.

It is important to account for seasonality in time series modeling because if seasonality is present in the data and not accounted for, it can lead to biased parameter estimates and inaccurate forecasts. By incorporating seasonality in the model, we can improve the accuracy of our forecasts and better understand the underlying patterns in the data.

3.3 XGBoost Regressor

XGBoost (eXtreme Gradient Boosting) is an ensemble learning algorithm that combines multiple decision trees to make predictions. The XGBoost algorithm works by iteratively adding decision trees to the model, where each tree corrects the errors of the previous tree. This process continues until a stopping criterion is reached, such as the maximum number of trees or a threshold for the improvement in performance.

Here are the basic steps involved in training an XGBoost regressor:

Initialize the model: XGBoost starts with a single decision tree with a single leaf node. The value of this leaf node is the mean of the target variable.

Calculate the residuals: The predicted values from the previous tree are subtracted from the actual target values to calculate the residuals. Train a new tree: A new decision tree is trained on the residuals. This tree tries to predict the residual values based on the input features.

Add the new tree to the model: The new tree is added to the model, and the predictions from all the trees in the model are summed to obtain the final prediction.

Repeat: Steps 2-4 are repeated for a specified number of iterations (i.e., the maximum number of trees) or until the performance improvement falls below a threshold.

Prediction: Once the model is trained, it can be used to make predictions on new

data by passing the input features through all the trees in the model and summing the predictions.

Overall, XGBoost is a powerful machine learning algorithm that can be used for a wide range of regression and classification problems. Its ability to handle missing values, regularization, and feature importance ranking makes it a popular choice for many data scientists and machine learning practitioners.

4. RESULTS

4.1 ARIMA

The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots are commonly used to determine the appropriate values of p and q respectively in an ARIMA model.
Alternatively we can use Auto ARIMA to determine the values of (p,d,q).The "auto" in Auto ARIMA refers to the fact that it automatically selects the best combination of ARIMA parameters (p,d,q) based on the time series data. After using Auto ARIMA we get (p,d,q) = (3,0,1). Here d=0 as our data is stationary.
Here we get the graph of actual vs predicted value on test data as

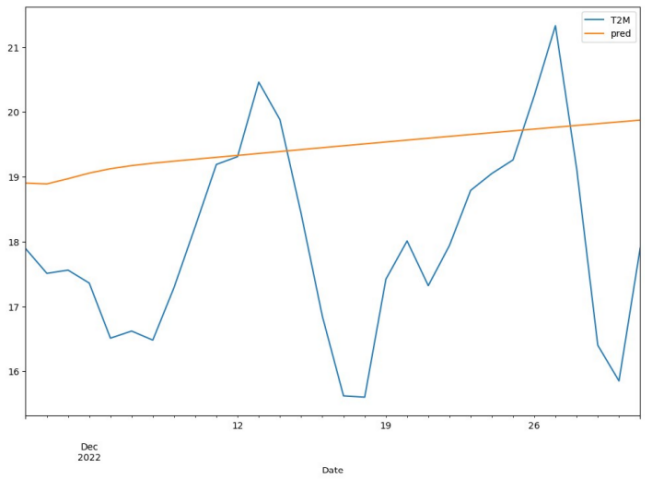


Fig 4.1

The result obtained is very poor due to the weak correlation between the current and lagged value.
Since the pattern of the data repeats, we use SARIMA with seasonal frequency 365. Seasonal ARIMA (SARIMA) models can be computationally intensive, particularly when dealing with high-frequency time series data such as daily data with a seasonal period of 365. This is because SARIMA models need to estimate a large number of parameters, which can require significant computational resources and time.
In addition, the seasonal period of 365 implies that there are a large number of lags that need to be considered for each season, leading to a very large number of potential model specifications that need to be evaluated. This can increase the computational time needed

to fit the model.

4.2 XGBoost

In this model we split the dataset into a training set and a testing set. The training set is used to train the XGBoost model, while the testing set is used to evaluate its performance. We consider the data from 1st January 2017 upto 26th November 2022 as train_data and the data after 26th November 2022 are consider as testing_data. Here we predict the temperature using only the previous temperature value. Figure 4.2 represents the results obtained.

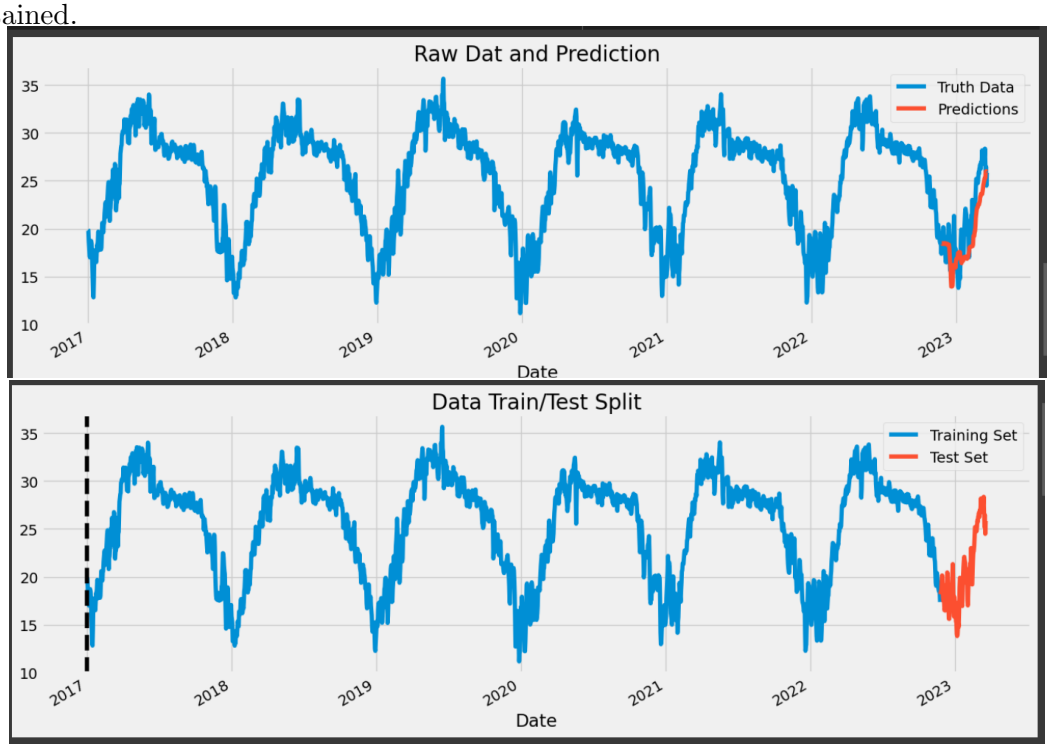


Fig 4.2

4.3 RNN

We have the data set with 8 features and 2265 input data.We split the data set into 2000 x_train data and 265 x_test data. Then we trained the data set with x_train shape (2000,3,8) and x_test shape (265,3,8). Training a LSTM neural network architecture with batch size 10 and 300 epochs we get a loss equal of 0.0014. Figure 4.3 shows the loss with varying epochs

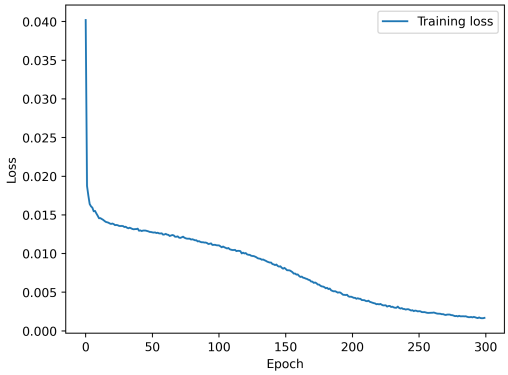
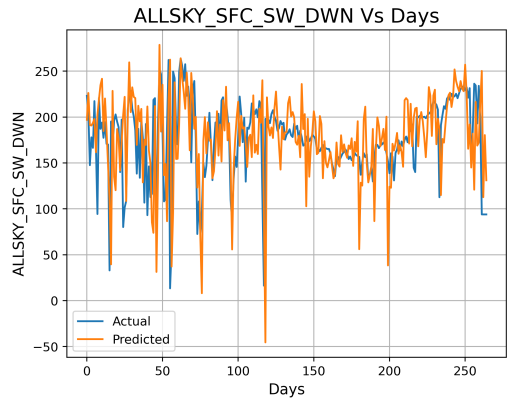
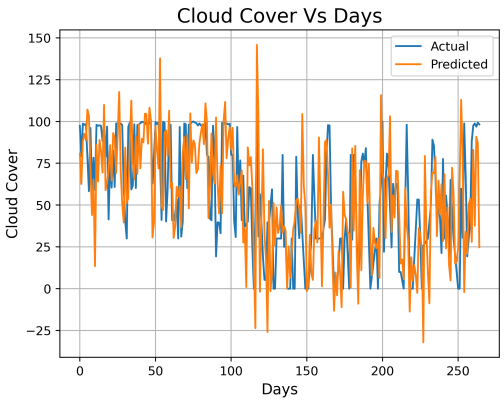
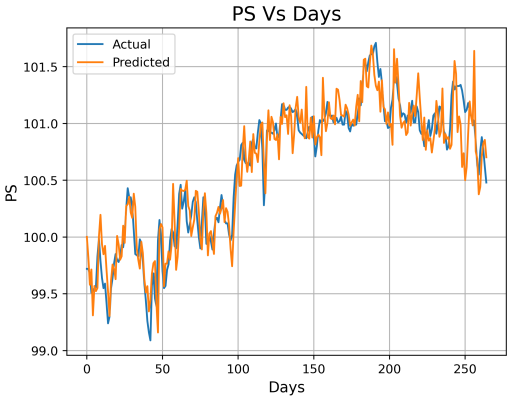


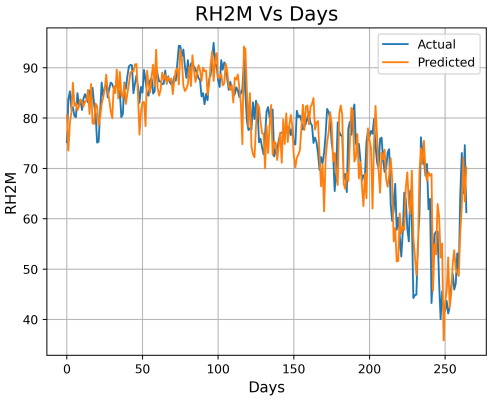
Fig 4.3

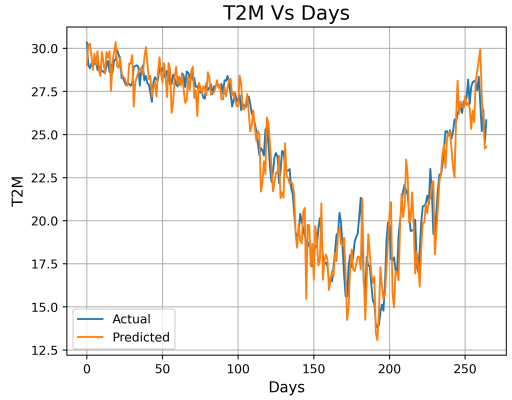
The following graphs show the predictions of all the 8 features and their actual values.

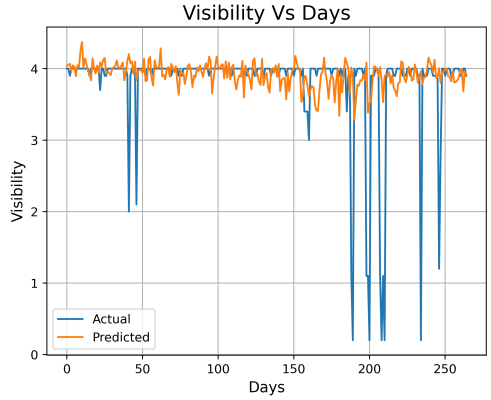












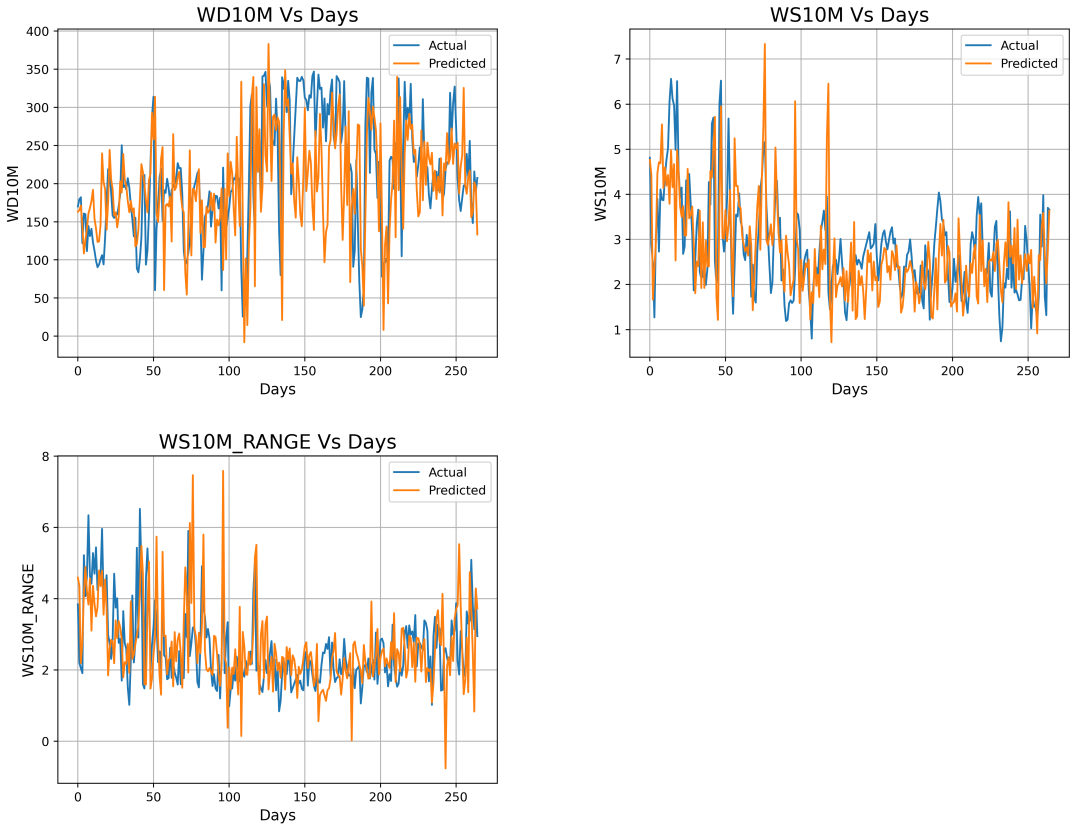


Fig 4.4

To further improve the predictions we train the model with a different input size. We now split the data set into 2150 x_train and 115 x_test data. After training an LSTM model with batch size 10 and 500 epochs we get a loss equal to 0.000488. The new loss vs epoch graph is shown in Figure 4.5

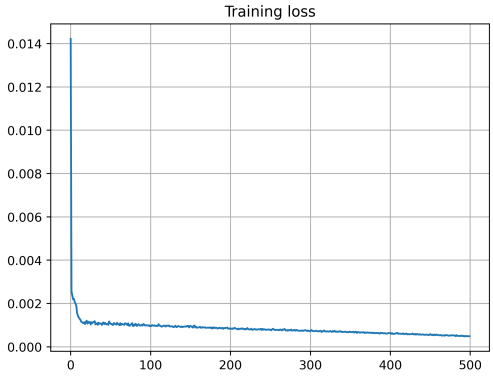


Fig 4.5 Loss Function

Prediction of temperature using all the eight features is as follows

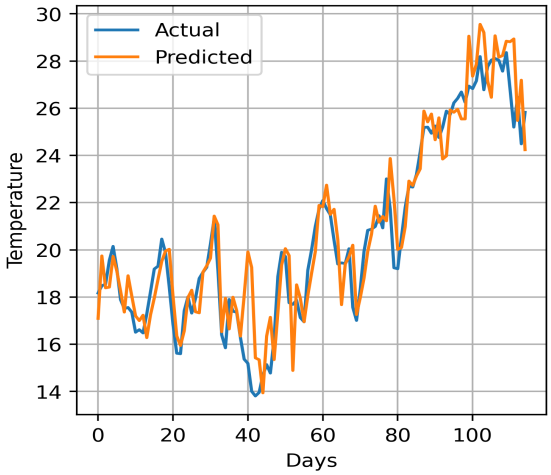


Fig 4.6

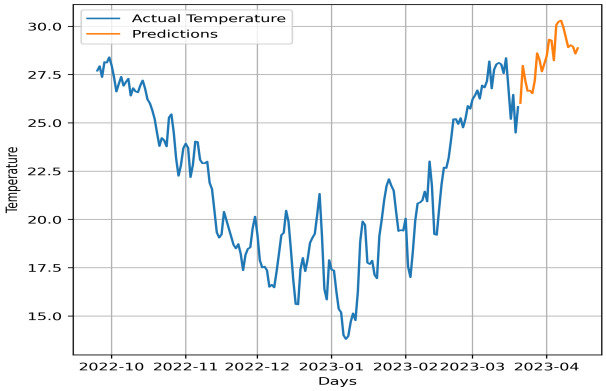


Fig 4.7 Predicted temperature from 21th march to 14th April using RNN (LSTM) model

4.4 Weather condition classification

Weather classification can be done with the help of various techniques. In this project, we have implemented the following models

Index	Model Name	Test Accuracy(%)
1	Gaussian Naive Bayes model	78
2	Support Vector Classifier	91
3	Decision Tree Classifier	83
4	Random Forest Classifier	86
5	Linear Discriminant Analysis	84

Tab. 4.1: Models and Accuracies

Table 4.2 shows the predictions made for the next 25 days using LSTM and LDA.

Date	Conditions	Average Temperature
2023-03-21	Overcast	25.56
2023-03-22	Clear	28.64
2023-03-23	Partially cloudy	28.27
2023-03-24	Partially cloudy	28.03
2023-03-25	Partially cloudy	28.90
2023-03-26	Overcast	28.81
2023-03-27	Partially cloudy	29.02
2023-03-28	Partially cloudy	30.88
2023-03-29	Partially cloudy	28.97
2023-03-30	Partially cloudy	28.40
2023-03-31	Partially cloudy	27.36
2023-04-01	Clear	28.05
2023-04-02	Clear	28.43
2023-04-03	Clear	29.20
2023-04-04	Partially cloudy	29.50
2023-04-05	Partially cloudy	29.78
2023-04-06	Partially cloudy	29.72
2023-04-07	Partially cloudy	28.74
2023-04-08	Partially cloudy	29.54
2023-04-09	Partially cloudy	30.40
2023-04-10	Partially cloudy	30.18
2023-04-11	Partially cloudy	30.70
2023-04-12	Partially cloudy	30.08
2023-04-13	Partially cloudy	29.96
2023-04-14	Partially cloudy	29.47

Tab. 4.2: Weather classification results

4.5 Conclusions

This project involved the study and implementation of advanced predictive models, including XGBOOST, ARIMA and RNN, to forecast future temperature trends of Kharagpur region. Following a rigorous evaluation of all three models, RNN demonstrated superior performance in predicting temperature patterns from 21.03.2023 to 14.04.2023.

Moreover, our classification efforts involved a comprehensive analysis of various classifiers, such as GNB classifier, support vector classifier, Decision tree classifier, Random forest classifier, and Linear discriminant analysis. Factors such as the complexity of the atmosphere, rapidly changing weather conditions, limited understanding of certain weather phenomena, and climate change make weather prediction a complex and challenging field. In the future, we might add other data sources to our model, such as satellite imagery, and expand our model to multiple regions.

REFERENCES

[1] S. Elsworth and S. Guttel, "Time Series Forecasting Using LSTM Networks: A Symbolic Approach", arXiv:2003.05672v1 [cs.LG] 12 Mar 2020.

[2] A. Kocharekar, Bharat V. Nemade, Chetan G. Patil, Durgesh D. Sapkale, Prof. Sagar G. Salunke, "Weather Prediction for Tourism Application using ARIMA", IRJET, vol 6, 3586-3590.

[3] Mehmet Tektas, "Weather Forecasting Using ANFIS and ARIMA MODELS. A Case Study for Istanbul", Environmental Research, Engineering and Management, 2010. No. 1(51), P. 5 – 10