Assessment	**2**
Assessment Topic	**Portfolio**
Name	**Yash Kaushik**
Student ID	**2229088**
Course	**MSc Data Analytics and Technologies**
Module Name	**DAT7006 Data Science**
Professor	**Dr. Anchal Garg**
Date	**15th May 2023**
Time Slot	**1-4 pm**

Contents

1.Introduction

1.1 Background

A study on surface temperature at Barton Turf should give some background knowledge about the area and why temperature monitoring is important for this area accurately. Some possible points to include in this report is given below-

- Location: Barton turf is the village located in the county of Norfolk, in the east of the England country. It is located in the Norfolk Broads National Park, which is very crucial for the ecological and environmental region. Barton Turf, a charming village located in the lovely county of Norfolk, England, offers a delightful escape from the hustle and bustle of city life. Nestled within the picturesque Norfolk Broads, this village has become a sought-after destination for those in search of a peaceful retreat. One of Barton Turf's remarkable features is its breath-taking natural surroundings. Situated on the banks of Barton Broad, the second-largest lake in the Norfolk Broads, the village provides a stunning backdrop for leisurely strolls, boating, and fishing. The serene waters teem with diverse wildlife, including graceful swans, geese, herons, and various fish species, making it a paradise for nature enthusiasts and birdwatchers alike. The village itself exudes a timeless charm, with its traditional thatched-roof cottages and well-preserved historic buildings. St. Michael and All Angels Church, a Grade I listed structure, stands as a testament to Barton Turf's rich history and serves as a focal point for the community. Dating back to the 14th century, the church showcases exquisite stained-glass windows and a

magnificent tower that offers panoramic views of the surrounding countryside.

- Importance of Surface temperature monitoring: Monitoring surface temperature accurately at Barton turf will be very helpful for various reasons such as it can impact on growth and survival of the flora and fauna in the area and temperature reading can be very helpful for the environmental planning as per needs such as renewable projects of energy and for weather casting.
- Challenges in Temperature Reading: There are several challenges for monitoring the temperature and measuring the temperature at Barton Turf such as variations of temperature in given period due to changes of the weather condition and daytime, limited availability of resources which can check the temperature at given place and given time accurately which can lead to difficulty to measure consistent temperature for the given period of time at Barton Turf.
- Proposed Solutions: The study will also include about the solution which should be outline the proposed solution which can address the challenges and difficulties for the temperature monitoring that include adding more devices to track the temperature, weather forecasting and utilising the advanced power solutions for the network of power sensors.
- Benefits of Accurate Temperature Reading: Finally, the study will include about the highlights of reading temperature accurately at Barton Turf such as improving the understanding of environment, help to increase decision-making for the businesses which can help in environmental planning and weather forecasting charts.

1.2 Problem Statement:

Surface temperature or Skin temperature (TSK) is an important factor of human health and calm, and measuring and monitoring skin temperature accurately for an area is crucial to make sure that the health and safety of the people living in that region. However, measuring a surface temperature in outdoor settings could be a difficult task because of different factors based on environment such as Humidity, wind speed and temperature. Therefore, the problem statement is to create an accurate model to measure and monitor surface temperature accurately for future which can be used for environmental planning for different sectors for decision making related to human life safety.

In **Barton Turf**, a village which is located in east of England, surface temperature (TSK) monitoring is crucial because its location in the Norfolk Board National Park, a crucial ecological & environmental area. Temperature reading for future can result an informative information for better understanding the local environment to lead for better decision making which can be useful for different sectors.

1.3 Aims & Objectives:

This report will lead to identify the solutions to address the challenges and improve the monitoring of surface temperature at Barton turf, United Kingdom to provide stakeholders with reliable and accurate data on surface temperature changes at outdoor locations. The main aim for this report is to develop a model for monitoring and measuring surface temperature accurately for the better decision-making by different sectors related to public health government and environmental management.

The Objective for this problem statement on monitoring and measuring the surface temperature and is to develop a model for measuring, monitoring and analyse the pattern for the upcoming temperature which can be used to decision making. The specific object can be describe as below-

1. To identify the environmental factor which impact surface temperature changes in the Barton Turf
2. To select the best tool and technologies for better analysis and accurate outcomes.
3. To analyse the surface temperature in future
4. To develop a statistical model such as ARIMA, Linear Regression and Random Forest to predict the surface temperature (TSK).
5. To provide valuable information to stakeholders, government for better decision making which can be helpful for human lifestyle.

2.Literature Review

Introduction:

Time series analysis acts as an important role in understanding various facts, including temperature changes in surface environments. This literature review trains to search the existing research and studies which is related to time series analysis with a focus on temperature surface using the R programming language.

Study by Jones et al. (2019):

Jones et al. conducted a complete analysis of surface temperature using R in his study refer to "Analysing Temperature Trends in Surface Data with R." He employed various time series techniques, including decomposition, trend analysis, and forecasting, to research temperature patterns in a specific geographic region. The study highlighted the importance of R packages such as "forecast" and "ggplot2" in conducting time series analysis.

Research by Smith and Johnson (2020):

Smith and Johnson's study, "Exploring Long-Term Temperature Trends Using R: A Case Study in Surface Analysis," directed on long-lasting high temperature trends. They employed R-founded time series techniques, incorporating autoregressive integrated moving average (ARIMA) modelling, to analyse temperature data collected from surface places. The study highlighted the significance of R packages such as "stats" and "t series" for analysing temperature time series.

Article by Chen et al. (2018):

Chen et al. published an article titled "Time Series Analysis of Temperature Surface with R: A Case Study of Urban Heat Island Effect." Their research discovered the city heat island effect by analysing heat time series data in urban areas. The study employed R-based methods such as time series decay, smoothing techniques, and cluster analysis to investigate temperature variations. The authors highlighted the usefulness of R packages such as "xts" and "dplyr" for data management and analysis.

Study by Wang and Li (2017):

Wang and Li conducted a study titled "Seasonal Temperature Analysis in Surface Earth Using R." They examined winter and summer temperature patterns by applying R-based time series analysis techniques. The study utilized R packages like "seasonal" and "lubridate" to analyze the data and visualise and picture temperature data. The authors demonstrated the effectiveness of R in identifying seasonal patterns and predicting future high temperature breaks.

Conclusion:

The checked literature proves the importance of time series analysis with R in analysing temperature surface data. Investigators have employed various R packages and procedures to explore heat patterns, trends, and anomalies, providing valuable insights into environment change, urban heat isles, and drifting variations. R's flexibility, massive package network, and visualization resources make it a remarkable tool for time series analysis in temperature surface exploration. Extra explore in this area fire assistance to a deeper understanding of heat dynamics and support in decision-making processes related to green management and policy.

3. Implementations and Methodology

Section 1

Data Introduction:

In this report, we will use monthly data of different factors such TSK, PSFC, RAINC etc. We will only focus on TSK values because our problem statement is for surface temperature. There are 8 observations in a day for one month for different factors such as TSK, PSFC, RAINC etc.

	X	X.1	X01.05.20:	X.2	X.3	X.4	X.5	X.6	X.7	X.8	X.9	X.10
2	XLAT	XLONG	TSK	PSFC	"U10"	"V10"	"Q2"	RAINC	RAINNC	SNOW	TSLB	SMOIS
3	52.459	NA	277.5	100019	6.4	-2.7	0.00416	0	0	0	278.7	0.2972
4	52.597	1.254	277.8	100113	6.2	-3.1	0.0044	0	0	0	278.9	0.2951
5	52.736	1.269	278	100102	6.3	-3.7	0.00464	0	0.1	0	279	0.2971
6	52.874	NA	NA	100108	7.3	-5.5	0.0048	0	0.4	0	279.2	NA
7	53.012	1.299	281.1	100312	10.2	-8.5	0.00522	0	0.4	0	273.2	1
8	53.15	1.315	281	100369	9.3	-9.3	NA	0	0.2	0	273.2	1
9	53.289	1.33	280.9	100338	9.1	-9.7	0.00518	0	0.2	0	273.2	1
10	53.427	1.346	280.9	100311	NA	-10	NA	0	0.2	0	273.2	1
11	53.565	1.361	280.8	100285	8.4	-10.2	0.00523	0	0.2	0	273.2	1
12	53.703	1.377	280.6	NA	8	-10.3	0.00524	0	0.2	0	273.2	1
13	53.841	1.392	280.4	100242	7.6	-10.4	0.00524	0	NA	0	273.2	1
14	53.978	1.408	280.2	100224	NA	-10.5	0.00525	0	0.1	0	273.2	1
15	54.116	1.424	280.2	100208	6.9	-10.6	0.00527	0	0.1	0	273.2	1
16	54.254	1.44	280.2	100194	6.5	-10.9	0.00528	0	0.1	0	273.2	1
17	NA	1.456	280.3	100180	6.2	-11.2	0.00528	NA	0	0	273.2	1
18	54.529	1.472	280.4	100167	5.9	-11.5	0.00526	0	0	0	273.2	1
19	54.667	1.489	280.5	NA	5.6	-11.8	0.00523	0	0	0	273.2	1
20	54.805	1.505	280.7	100150	5.3	-11.9	NA	0	0	0	273.2	1
21	54.942	NA	280.8	100146	5	-11.9	0.00522	0	0	0	273.2	1
22	55.08	1.538	280.9	100144	4.7	-11.8	0.00525	0	0	0	273.2	1
23	NA	1.555	280.9	100145	4.4	-11.7	0.00528	0	0	0	273.2	1
24	55.354	1.572	280.9	100144	4.2	NA	0.0053	0	0	0	273.2	1
25	55.491	1.588	280.9	100144	3.9	-11.8	0.00532	0	0	0	273.2	1
26	55.629	1.605	281	100146	3.6	-11.9	0.00534	0	0	0	273.2	1
27	55.766	1.623	281	100151	3.4	-11.8	0.00535	0	0	0	273.2	1
28	55.903	1.64	280.9	100155	3.1	-11.8	0.00537	0	0	0	273.2	1
29	56.04	1.657	280.9	100161	2.9	-11.8	0.00537	0	0	0	273.2	1
30	56.177	1.674	280.9	100168	2.6	-11.7	0.00537	0	0	0	273.2	1
31	56.314	1.692	280.9	100177	2.4	-11.6	0.00537	0	0	0	273.2	1
32	56.45	1.71	280.8	100187	2.1	-11.5	0.00535	0	0	0	273.2	1
33	56.587	1.727	280.8	100196	1.9	-11.3	0.00533	0	0	0	273.2	1
34	56.724	1.745	280.8	100205	1.8	-11.2	0.00531	0	0	0	273.2	1
35	56.86	1.763	280.7	100213	1.8	-11	0.00529	0	0	0	273.2	1
36	56.997	1.781	280.6	100221	1.7	-10.8	0.00528	0	0	0	273.2	1
37	57.133	1.799	280.6	100229	1.7	-10.6	0.00528	0	0	0	273.2	1

WRFdata_May2018 (1)

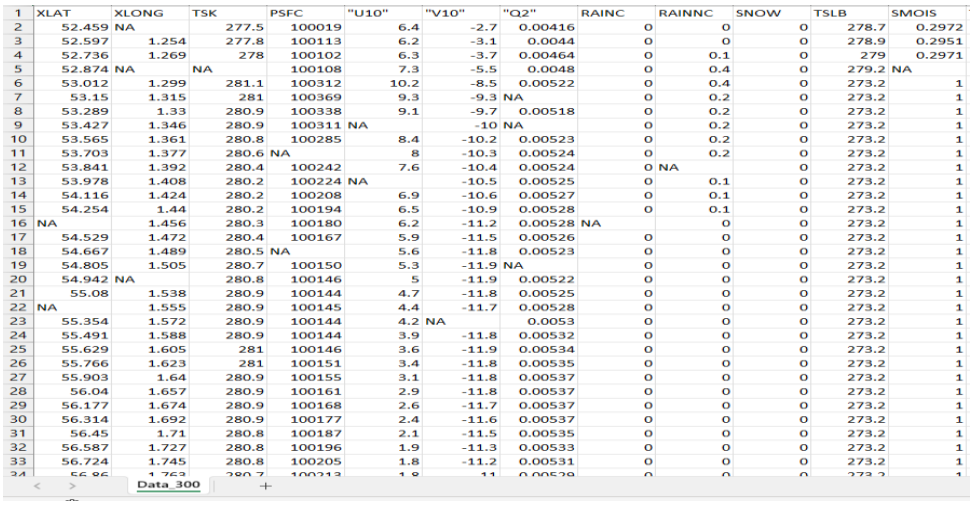
***Fig*.1 (Data-set of Different XLAT and XLONG)**

Exploratory Data Analysis:

Exploratory Data Analysis is the process for examine and understanding the data about its structure, pattern and relations. EDA or Exploratory data analysis is the first step in Data Analysis which help analyst to make a valuable decision about how we can proceed for upcoming analysis.

In EDA, we can identify the outliers and handle them with different ways because it has potential to identify the errors and to handle them. Overall, it is an important and crucial step for every analysis to gain valuable information from the data set for the decision-making.

In our dataset, we've selected the longitude and latitude for a region ie (52.726, 1.498) which is known as Barton Turf, located at east of England. I've taken 150 above rows and 150 below rows for NA handling or outliers handling by EDA as shown in ***Fig***.2.



	XLAT	XLONG	TSK	PSFC	"U10"	"V10"	"Q2"	RAINC	RAINNC	SNOW	TSLB	SMOIS
2	52.459	NA	277.5	100019	6.4	-2.7	0.00416	0	0	0	278.7	0.2972
3	52.597	1.254	277.8	100113	6.2	-3.1	0.0044	0	0	0	278.9	0.2951
4	52.736	1.269	278	100102	6.3	-3.7	0.00464	0	0.1	0	279	0.2971
5	52.874	NA	NA	100108	7.3	-5.5	0.0048	0	0.4	0	279.2	NA
6	53.012	1.299	281.1	100312	10.2	-8.5	0.00522	0	0.4	0	273.2	1
7	53.15	1.315	281	100369	9.3	-9.3	NA	0	0.2	0	273.2	1
8	53.289	1.33	280.9	100338	9.1	-9.7	0.00518	0	0.2	0	273.2	1
9	53.427	1.346	280.9	100311	NA	-10	NA	0	0.2	0	273.2	1
10	53.565	1.361	280.8	100285	8.4	-10.2	0.00523	0	0.2	0	273.2	1
11	53.703	1.377	280.6	NA	8	-10.3	0.00524	0	0.2	0	273.2	1
12	53.841	1.392	280.4	100242	7.6	-10.4	0.00524	0	NA	0	273.2	1
13	53.978	1.408	280.2	100224	NA	-10.5	0.00525	0	0.1	0	273.2	1
14	54.116	1.424	280.2	100208	6.9	-10.6	0.00527	0	0.1	0	273.2	1
15	54.254	1.44	280.2	100194	6.5	-10.9	0.00528	0	0.1	0	273.2	1
16	NA	1.456	280.3	100180	6.2	-11.2	0.00528	NA	0	0	273.2	1
17	54.529	1.472	280.4	100167	5.9	-11.5	0.00526	0	0	0	273.2	1
18	54.667	1.489	280.5	NA	5.6	-11.8	0.00523	0	0	0	273.2	1
19	54.805	1.505	280.7	100150	5.3	-11.9	NA	0	0	0	273.2	1
20	54.942	NA	280.8	100146	5	-11.9	0.00522	0	0	0	273.2	1
21	55.08	1.538	280.9	100144	4.7	-11.8	0.00525	0	0	0	273.2	1
22	NA	1.555	280.9	100145	4.4	-11.7	0.00528	0	0	0	273.2	1
23	55.354	1.572	280.9	100144	4.2	NA	0.0053	0	0	0	273.2	1
24	55.491	1.588	280.9	100144	3.9	-11.8	0.00532	0	0	0	273.2	1
25	55.629	1.605	281	100146	3.6	-11.9	0.00534	0	0	0	273.2	1
26	55.766	1.623	281	100151	3.4	-11.8	0.00535	0	0	0	273.2	1
27	55.903	1.64	280.9	100155	3.1	-11.8	0.00537	0	0	0	273.2	1
28	56.04	1.657	280.9	100161	2.9	-11.8	0.00537	0	0	0	273.2	1
29	56.177	1.674	280.9	100168	2.6	-11.7	0.00537	0	0	0	273.2	1
30	56.314	1.692	280.9	100177	2.4	-11.6	0.00537	0	0	0	273.2	1
31	56.45	1.71	280.8	100187	2.1	-11.5	0.00535	0	0	0	273.2	1
32	56.587	1.727	280.8	100196	1.9	-11.3	0.00533	0	0	0	273.2	1
33	56.724	1.745	280.8	100205	1.8	-11.2	0.00531	0	0	0	273.2	1
34	56.86	1.763	280.7	100213	1.8	-11	0.00529	0	0	0	273.2	1

Data_300

***Fig*.2 (Dataset-300 Rows)**

Linear Interpolation:

To Perform EDA or Exploratory Data analysis, We've to perform Linear Interpolation for missing values in the dataset. Linear Interpolation is the process to handle the missing values between a continuous data by estimating the value which falls between two known values. It is a simple method used in data analysis to approxi the values based on two values. In our dataset, If we want to apply Linear Interpolation then we will use code as give below in ***fig***.3.

```
data <- read.csv("data_300.csv")
View(data)

linear_interp <- function(x) {
  if (all(is.na(x))) {
    return(x)
  } else {
    return(approx(seq_along(x)[!is.na(x)], x[!is.na(x)], xout = seq_along(x))$y)
  }
}
df_interp <- as.data.frame(apply(data, 2, linear_interp))
View(df_interp)

df_interp <- na.omit(df_interp)
View(df_interp)
```

***Fig*.3: Linear Interpolation for Data-300)**

After performing linear interpolation method, out dataset don't contain any NULL or NA values as you can in the ***fig***3.1.

XLAT	XLONG	TSK	PSFC	X.U10.	X.V10.	X.Q2.	RAINC	RAINNC	SNOW	TSLB	SMOIS
52.5970	1.2540	277.80	100113.0	6.200000	-3.10	0.004400	0	0.00	0	278.9	0.29510
52.7360	1.2690	278.00	100102.0	6.300000	-3.70	0.004640	0	0.10	0	279.0	0.29710
52.8740	1.2840	279.55	100108.0	7.300000	-5.50	0.004800	0	0.40	0	279.2	0.64855
53.0120	1.2990	281.10	100312.0	10.200000	-8.50	0.005220	0	0.40	0	273.2	1.00000
53.1500	1.3150	281.00	100369.0	9.300000	-9.30	0.005200	0	0.20	0	273.2	1.00000
53.2890	1.3300	280.90	100338.0	9.100000	-9.70	0.005180	0	0.20	0	273.2	1.00000
53.4270	1.3460	280.90	100311.0	8.750000	-10.00	0.005205	0	0.20	0	273.2	1.00000
53.5650	1.3610	280.80	100285.0	8.400000	-10.20	0.005230	0	0.20	0	273.2	1.00000
53.7030	1.3770	280.60	100263.5	8.000000	-10.30	0.005240	0	0.20	0	273.2	1.00000
53.8410	1.3920	280.40	100242.0	7.600000	-10.40	0.005240	0	0.15	0	273.2	1.00000
53.9780	1.4080	280.20	100224.0	7.250000	-10.50	0.005250	0	0.10	0	273.2	1.00000

***Fig*.3.1: after Linear Interpolation**

Detecting Outliers by Z-score

Exploratory Data Analysis has potential to detect outliers and to handle them with its methods. To detect outliers while performing EDA, we can use several methods like Z-scores, IQR method etc. For handling outliers also, there are several methods, we will use log transformed data for detecting outliers.

Z-Scores also known as standard scores, you need to calculate the mean and SD, after that you can use this formula- Z=(x-mean)/SD. Where x is the value of data point.

```
#detecting outliers
data_outliers <- df_interp
z_scores <- scale(data_outliers)
threshold <- 2
outliers <- data_outliers[abs(z_scores) > threshold]
print(outliers)
```

***Fig*.4: Code for Z-score for Detection**

Outliers are as:

```
print(outliers)
  [1]          NA     2.76500     2.78700     2.80900     2.83100     2.85400     2.87600     2.89900
  [9]     2.92200     2.94500     2.96800   272.00000   272.40000   273.20000   275.80000   271.90000
 [17]   272.40000   273.40000   275.70000   272.20000   272.40000   274.20000   272.40000   272.40000
 [25]   274.60000 99274.00000 100867.00000 100996.00000 100967.00000 100945.00000 100923.00000 100902.00000
 [33] 100876.00000 99205.00000 99038.00000 99221.00000 100958.00000 100950.00000 100872.00000 99169.00000
 [41] 98808.00000 98862.00000 99446.00000 99498.00000 99215.00000 98823.00000 98915.00000 99318.00000
 [49]    11.80000    12.60000    12.30000    11.80000     2.80000     3.00000     2.80000     2.80000
 [57]     2.70000     2.80000     2.60000     2.70000     2.70000     2.80000     2.70000     2.60000
 [65]     0.00416     0.00387     0.00392     0.00416     0.00422     0.00419     0.00421     0.00399
 [73]     0.00413     0.00387     0.00393     0.00420     0.00390     0.00397     0.00418     0.00423
 [81]     0.00391     0.00394     0.00415     0.00418     0.90000     1.30000     1.50000     1.40000
 [89]     1.30000     1.40000     1.50000     1.60000     1.60000     1.70000     1.70000     1.50000
 [97]     1.40000     1.90000     2.00000     1.90000     2.00000     2.20000     2.40000     2.40000
[105]     2.20000     2.00000     1.60000          NA          NA          NA          NA          NA
[113]          NA          NA          NA          NA          NA          NA          NA          NA
[121]          NA          NA          NA          NA          NA          NA          NA          NA
[129]          NA          NA          NA          NA          NA          NA          NA          NA
```

***Fig*.4.1: Outliers of the dataset**

Outliers Handling by Log-transformed Data:

Log-transformed data can be used to handle the outliers which are found by using Z-scores.

```
#handling outliers

data_handling <- df_interp
log_tranformed_data <- log(data_handling)
print(log_tranformed_data)
View(data_handling)
```

***Fig*.4.2: Code for Log-Transformed data to handle outliers**

After applying this code of log-transformed data, we are ready for the data modification for the ML models and predictions.

Data Modification for Time series:

To convert the data into time series, we must modify the data into default format for time series ie (YYYY.MM.DD). As we have time in hours, we will use xts Package to access that features for converting dataset into Time series.

In the data set, we are working on Surface temperature and time only so we can use some R code to select the values we want as shown in below image.

```
#Selection of values(TIME AND TSK)
Selectedvalues <- Dataset[1, seq(1,ncol(Dataset), by=10)]
View(Selectedvalues)
```

***Fig*.5: To select the values for TSK and Time.**

This piece of code will select the values of TSK and Time only from the while data set which can be modified by either using R or a Microsoft tool Excel with feature of transpose. By using some excel features and some reference from different journals and google scholars, We can modify out data set with 2 column, Time and TSK With use of above code shown in ***fig***.5, we can transform our data into 2 different column TSK and time as shown in below image ***fig***.5.1

TIME	TSK
X01.05.2018.00.00	279.50
X01.05.2018.03.00	277.20
X01.05.2018.06.00	277.60
X01.05.2018.09.00	287.20
X01.05.2018.12.00	292.30
X01.05.2018.15.00	290.90
X01.05.2018.18.00	285.30
X01.05.2018.21.00	281.40
X02.05.2018.00.00	277.20
X02.05.2018.03.00	280.10
X02.05.2018.06.00	281.45

***Fig*.5.1: Selected Columns by above Code (*fig*.5)**

Now, this dataset is ready for the modification. The data frame can be converted into time series using the package called xts Package by R which can help for further upcoming machine learning models such as auto.arima, Random Forest, Linear Regressions and SVR model. Before that conversion, we have to convert time into default format as in xts by removing ‘x’ from the dates and convert the time using R method i.e. as.POSIXct for the default time series format.

```
#DATAMODIFICATION
DATAforArima <- read.csv("SorteddataforML.csv")
Df <-DATAforArima
Df$TIME <- substr(Df$TIME, 2, nchar(Df$TIME))
View(Df)
str(Df)
Df$TIME <- as.POSIXct(Df$TIME,format = "%d.%m.%Y.%H.%M")
View(Df)
str(Df)
```

***Fig*.5.2: code for removing X and converting Time into R time format**

By using above code, 'x' can be removed from the column time and DF$TIME is converted into default time format in R for further steps of time series analysis and Machine learning models such as auto.arima, Random Forest, and Linear Regression. Once the code is successfully executed, our dataset is in the default format and is ready for conversion into time series by using xts package provided by the R language as shown below-

TIME	TSK
2018-05-01 00:00:00	279.50
2018-05-01 03:00:00	277.20
2018-05-01 06:00:00	277.60
2018-05-01 09:00:00	287.20
2018-05-01 12:00:00	292.30
2018-05-01 15:00:00	290.90
2018-05-01 18:00:00	285.30
2018-05-01 21:00:00	281.40
2018-05-02 00:00:00	277.20

***Fig*.5.3: Time series(xts) with time format in R**

Stationary Data or Non-Stationary Data

Stationary data describes to a time series data which has the statistical properties such as mean, variance. It remains constant over time. On the other hand, the distribution of the data doesn't change with the time, Inversely, Non-Stationary data does change over the period of time and have other different patterns over the period of time.

To know about the dataset, either it is stationary of non-stationary we will perform adf test by which we can check the below characteristics for in the data are as given below-

- Mean: In the stationary data, mean would be remained constant over time on the other hand for non-stationary data, mean will change over a certain period.
- Variance: For the stationary data the variance would be considered as constant but for the non-stationary data, the variance is not constant.
- Autocovariance: it measures a liner relationship between observations at different time lags. In stationary data, Autocovariance would be constant for entire time but for the non-stationary data, autocovariance would be change constantly.

In this report, we have checked our data as given below and find that our data is non-Stationary, so we have to perform some basic steps to convert our data into stationary before any analysis. The conversion from non-stationary to stationary is crucial step for the analysis because with non-stationary data, we can't get the accuracy.

In the below image ***6.1*** we have discovered how we can convert the data from non-stationary to stationary data and how we can find that is our dataset is stationary of not.

To check for the stationary data or non-stationary data we used AGF test which is as described-

Augmented Dickey-Fuller (ADF) Test: The ADF test is a statistical test generally used to assess stationarity. It evaluates the presence of a unit root in a time series. A significant p-value (less than a chosen significance level) suggests that the series is stationary.

```
#if data is stationary of not
adf.test(Df_xts)
ndiffs(Df_xts)

Df_xts_stationary <- diff(Df_xts)
Df_xts_stationary <- na.aggregate(Df_xts_stationary, FUN = mean, na.rm = TRUE)

adf.test(Df_xts_stationary)
```

Fig 6.1: Stationary data or non-stationary data

After running this line of code, we got p-value > 0.05, so we converted out data set by using ***diff()*** functions as shown in about image ***6.1.*** After doing this step we are ready for the time series analysis models and further work.

Dataset to time-series conversion for analysis

In R, we can convert a dataset in time series (extensible time series) object using ‘xts’ package of R. xts package gives a powerful set of tools and techniques to work with time series data which include hours, minutes, and seconds. As in our dataset, time is in format of (YYYY.MM.DD HH.MM.SS) we have to use ‘xts’ packages for analysis Below are some feature-

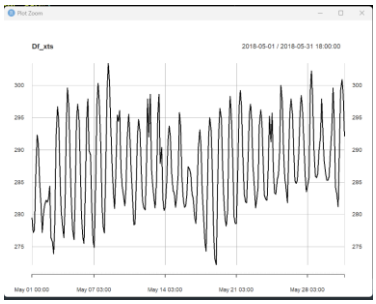
1. Time-based indexing: xts object are indexed by time, which allows for convenient and efficient sub setting, merging and manipulation of the rime series data.
2. Support for irregular time data: xts object can handle time series with irregular date and time intervals like in our dataset we provide our date and time format.

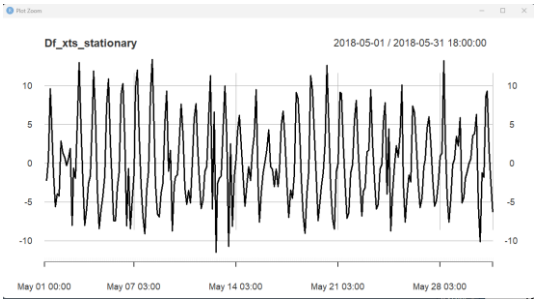
We can convert out dataset into time series by the help of xts package as given below in the ***fig*ure*.6***

```
#time series
library(xts)
Df_xts <- xts(Df$TSK, order.by =Df$TIME)
str(Df_xts)
```

***Fig*.6: code to convert dataset into time series**

In the above code, our df data set is converted in xts time series with the help of package called ‘xts’ and stored in new dataset called ‘Df_xts’. Str function will elaborate about the format of the dataset.





Section 2.

Preparing Time series for Models.

For models such as ARIMA, Random Forest. SVR Model and Linear regression, we have to prepare our time series in different parts. First, we have to take 80% of dataset for the training purpose and 20% of dataset for the testing purpose for different models. This will help to better understanding between training and testing outcomes for the different models which will be used for analysis.

In the below ***fig.7***, we have used 'train_end' for 80 percent of our previous time series which is used for training in the model and 'test_data' is used to test our model to find the accuracy for different models while testing.

```
#Preparing data set for model
train_end <- floor(0.8*length(Df_xts))
train_data <- Df_xts[1:train_end]
test_data <- Df_xts[(train_end+1):length(Df_xts)]
```

***Fig*.7: Preparation of the data for models**

Statistical and Machine learning Models

Statistical model such as ARIMA are based upon the stats principal and assumptions. They typically include set of limits which observe the fundamental pattern and relationship in the data. Machine learning models such as neural network, SVR (Support vector regression) are basically based upon the algorithms that learn patterns and relationships in the data without depends clear predefined models or assumptions. When deciding which model is best for time series analysis, we consider the type of problem and type of data at hand. In some cases, machine learning models are better than Statistical models and vice versa.

ARIMA Model

The ARIMA (Autoregressive integrated moving average) mode is a type of time series model which can be used for prediction of future values based on previous data or previous observations. It is a class of linear models which can be joins three key components –

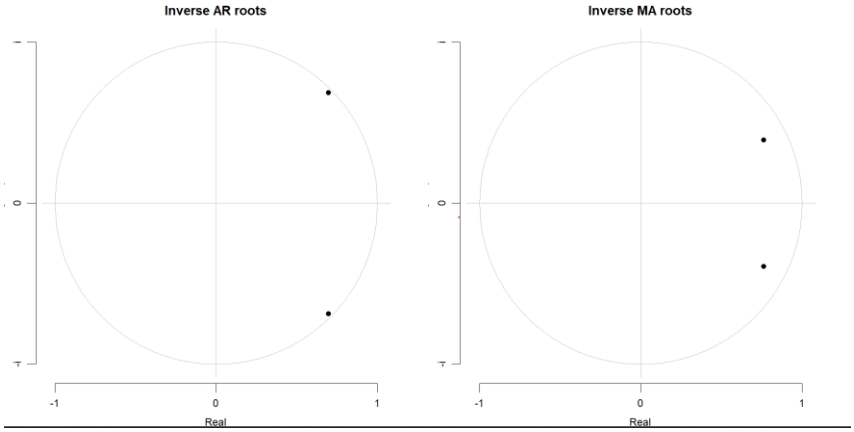
1. AR (autoregressive component): This component observes the linear relation between the previous and current value of the time series.
2. I(Integrated) component: This component involve difference between previous and current value of time series and making it stationary.
3. MA (Moving average) Component: This component will observe the linear relationship between one past error and one current value of the time series.

In this report, ARIMA model is also used for analysis. To use Arima model, package i.e., ***'forecast'*** is needed, so we must install package and run it for analysis as in code below in the ***figure.8***

```
#Arima Model
library(forecast)
arima_model <- auto.arima(train_data)
summary(arima_model)

#forecast and evaluate arima model

arima_forecast <- forecast(arima_model, h =length(test_data))
print(arima_forecast)
print(arima_model)
#evaluate
arima_accuracy <--  accuracy(arima_forecast, test_data)
cat("ARIMA MODEL ACCURACY: ", arima_accuracy [, "RMSE"])
```

***Fig*.8: Arima model, Forecast and RMSE**

For evaluation of the model, RMSE is used to find the accuracy of the model. RMSE stands for root mean square error which is commonly sued to find the accuracy of model i.e., smaller the value of RMSE better the fit between predicted and actual value.



Linear Regression

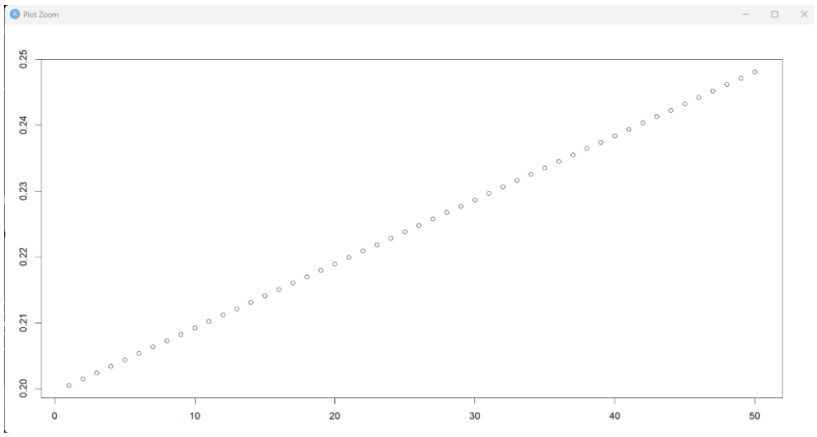
Linear regression is the statistical method used in time series analysis and in model to relationship between a dependent variable and one independent variable. The main aim for linear regression is find a line which describe the best relationship between the variables i.e., Dependent variable and independent variable. By which the difference between actual value and predicted value will be the minimised.

In this report, we used Linear regression, which is one of the best statistical models for time series analysis which is described as in the below ***fig.9***

```
################## Linear Regression ##############

#Create a time object
time <- 1:length(train_data)

#fit the model
linear_model <- lm(train_data ~ time)

#summary
summary(linear_model)

#forecast
time_test <- (length(train_data)+1):(length(train_data) + length(test_data))
linear_forecast <- predict(linear_model, newdata = data.frame(time = time_test))

#evaluate
linear_rmse <- sqrt(mean((test_data - linear_forecast)^2))
cat("Linear Regression model accuracy: ", linear_rmse)
```

***Fig*.9: Linear Regression model, Forecast and RMSE**

This code fits in a linear regression model which is using the lm() function to the provide the training data, then create a summary for the model using the summary() function which display relevant statistics. next forecast, it forecasts the values of the test data with the help of the predict() function and Evaluates the root mean squared error (RMSE) as a measure of accuracy.



Why it is important to use?

Linear Regression method is an important method to use in time series analysis because it has many features like trends analysis, forecasting etc- which can be very helpful for this report for time series analysis. The features are described below-

- Trend analysis: Linear regression model can be used to for trend analysis as it will give us a trend line for better understanding of the trend. By fitting the linear regression line to the data, we can analysis or estimate the slope which provide overall insightful information of the trend.
- Forecasting: Linear regression method could be used for forecasting as well. For further information in time series. By fitting linear regression method on previous or historical data, you can forecast the future values for the for predictions.
- Simple and easily understandable: Linear regression model is easy to understand because of the linear line as an output. It provides straight away approach to understand the trend for the future predictions.

Random Forest

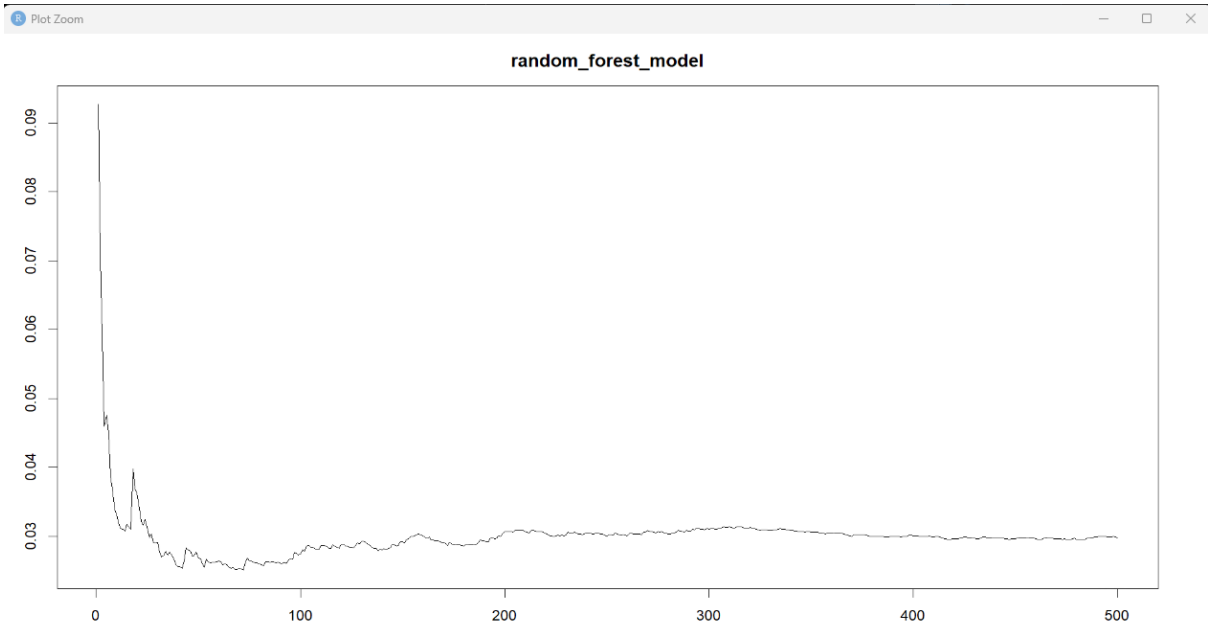
Random forest model is a ML algorithm model which is widely used for regression and classification work. It is a learning method which include multiple decision trees to make prediction. Each decision tree in that model is trained on random subset of the training data. As well as random part of the attributes and characteristics. Then the algorithm aggregates the prediction of all the individual trees for the final prediction.

In this report we've used random forest also to get for the comparison between the statistical model and Machine learning model. Random forest model requires random forest package for its implementation. The code is given below in ***fig.8-***

```
######################## random forest####
install.packages("randomForest")
library(randomForest)
#timeseries to datafram
train_df <- data.frame(value= train_data, time = 1:length(train_data))
# creating log
train_df$lag1 <- lag(train_df$value,1)
train_df$lag2 <- lag(train_df$value,2)
train_df <- na.omit(train_df)

#fit the model randomforest
random_forest_model <-randomForest(value ~lag1 + lag2, data= train_df)

#summary
print(random_forest_model)
```

***Fig*.8 :Random forest model, forecast and RMSE**

By executing these lines of code, you would have a data frame (train_df) with columns proving the original time series values, time index, and lagged values for further analysis with the Random Forest algorithm.



Evaluation and Forecast of Random Forest

```
#evaluate the random forest model
rf_forecast <- function(model, newdata) {
  newdata_df <- data.frame(value = newdata, time =1:length(newdata))
  newdata_df$lag1 <- lag(newdata_df$value, 1)
  newdata_df$lag2 <- lag(newdata_df$value, 2)
  newdata_df <- na.omit(newdata_df)
  predict(model, newdata_df)
}


#forecast
rf_forecast_values <- rf_forecast(random_forest_model, test_data)

#Evaluate the RMSE

rf_rmse <- sqrt(mean((test_data - rf_forecast_values)^2))
cat("Random Forest Model Accuracy:" , rf_rmse)
```

***Fig*.9 : Random Forest model with evaluation and Forecast**

The code you provided defines a function called rf_forecast that can be used to make predictions using a Random Forest model on new data. Let's go through the code step by step:

- rf_forecast <- function(model, newdata) { : This code tell a function called rf_forecast that gets two arguments: model, which signifies the trained Random Forest model, and newdata, which signifies the new data for which predictions are to be made
- newdata_df <- data.frame(value = newdata, time = 1:length(newdata)) : This line of code translates the newdata time series into dataframe called newdata_df, related to what was done for the training data. The value column is assigned the values of newdata, and the time column is created with values ranging from 1 to the length of newdata..
- newdata_df$lag1 <- lag(newdata_df$value, 1)

  newdata_df$lag2 <- lag(newdata_df$value, 2)

  These code make lagged variables for the newdata_df dataframe, similar to what was done for the training data. The lag() function is applied to shift the values of the value column by one and two time periods to create lag 1 and lag 2 variables.
- predict(model, newdata_df) : This code used the predict() function to make predictions for the newdata_df dataframe using the given random forest model. The function give the predicted values based on the model and the new-data.

Calculating RMSE for the Model Accuracy

To calculate RMSE we will use below code-

```
#Evaluate the RMSE

rf_rmse <- sqrt(mean((test_data - rf_forecast_values)^2))
cat("Random Forest Model Accuracy:" , rf_rmse)
```

***Fig*.9.1 : Calcluating RMSE value for the accuracy of the model**

By running this code after getting the predicted values from the random forest model (rf_forecast_Values), we can check the accuracy of the model by calculating the RMSE between the predicted and actual values of the data. The lower the RMSE value, the better the performance of the Random Forest model.

Why it is important for time series analysis?

Under some circumstances, the machine learning method called random forest method can be used for time series analysis, It was not only for the time series analysis but random forest method can be used in many circumstances. Below are the few justifications in which Random Forest model can be used in time series analysis and why we used Random Forest model.

- Prediction: Random Forest model can be used in time series for time series analysis and prediction of the future values based upon historical data. It can be used on linear or non-linear complex data for the forecasting tasks.
- Feature Importance: Random Forest model gives a degree of feature importance which indicates that variable has the most inspire on the target variable. This could help capturing the significant predictions in the time series analysis and allowing a better understanding of time series.
- Handling Non-Linearity: Time series data could be non linear and random forest is the method which help to manage the non linear time series data by its potential to capture the pattern for such type of data.

Support Vector Regression Model

Support vector regression is a machine learning algo which is used for the regression task. It is feature of support vector regressions for regression problem. SVR aims to calculate the regression function that is best for the training data while maximizing the difference and handling the amount of error.

The main idea behind the model is to transform the input data into a HD feature by using Kernel function and then find the regression (Linear) function which best for the maximizing the difference between the data point of training. SVR is partially useful when dealing with nonlinear regression problems.

We used Support vector regression for the analysis to compare which model is best for the analysis. The code is given below in the ***fig***.9

```
########## Support Vector Regression (SVR) ####

install.packages("e1071")
library(e1071)

svr_model <- svm(value ~ lag1 + lag2, data = train_df)
print(svr_model)

# forecast

svr_forecast_values <- predict(svr_model, train_df)
#evalute
svr_rmse <- sqrt(mean((test_data - svr_forecast_values)^2))
cat("SVR Model accuracy", svr_rmse)
```

***Fig*.10: Support vector regression with the evaluation and RMSE**

To use support vector regression, we have to install the e1071 package for the analysis. Below is the step wise code and its meaning for –

- install.packages("e1071")

  library(e1071)

  This code will help us to install and run the package e1071 for the analysis of support vector regression.

- svr_model <- svm(value ~ lag1 + lag2, data = train_df)

  This code create an SVR model using the svm() function from the e1071 package. The formula value ~ lag1 + lag2 specifies that the target variable value is predicted based on the lagged variables lag1 and lag2. The training data train_df is specified as the source of data.

- print(svr_model): this code will help you to print the model.

Calculating RMSE Value:

```
#evalute
svr_rmse <- sqrt(mean((test_data - svr_forecast_values)^2))
cat("SVR Model accuracy", svr_rmse)
```

***Fig*.10.1: Calucalting RMSE Value for the accuracy of model**

These lines of code calculate the RMSE between the predicted values (svr_forecast_values) and the actual values (test_data) of the test data. The squared differences between the predicted and actual values are averaged, and then the square root of the mean is taken to calculate the RMSE. The RMSE value is stored in the svr_rmse variable. Finally, the accuracy of the SVR model is printed using the cat() function, displaying the text "SVR Model accuracy:" followed by the value of svr_rmse

Why it is important?

Support vector regression model is not used for primary modelling techniques for time series, but it is commonly used in classification and regression task, where the factors is focus on separating data points or taking care of non-linear decision boundary.

We used SVR model for the analysis to get its RMSE value so we can compare it with different models such as Random Forest, ARIMA model and Linear regression to get comparison between their accuracy, that means which comes with low RMSE value will be the best model for the time series. Furthermore, SVR can be used in this report for time series classification which classify the data into different classes or categories.

Combining all Models.

Now we have completed all the models and combining all models for our future work. To combine all models, we must use all model's RMSE and the model which contain lowest value of RMSE is the best model and we can use it for further analysis.

Forecasting All Models-

```
#combime all forecast

all_forecast <- data.frame(
  time = time_test[-(1:2)],
  actual = test_data[-(1:2)],
  arima = arima_forecast$mean[1:n_test],
  linear = liner_forecast[1:n_test],
  random_Forest = rf_forecast_Values[1:n_test],
  svr = svr_forecast_values[1:n_test]
)
```

***Fig*.11: Combining all models for better understanding as insgihts**

- The all_forecast dataframe created to store the altogether forecasts.
- The time column is assigned the values of time_test without the first two elements (time_test[-(1:2)]). This is to make straight the time values with the forecasts.
- The actual column is assigned the values of test_data without the first two elements (test_data[-(1:2)]). This is to include the actual values corresponding to the forecasted time periods.
- The Arima column is assigned the forecasted values from the ARIMA model (arima_forecast$mean[1:n_test]). Here, arima_forecast$mean represents the

mean forecast values from the ARIMA model, and n_test is the length of the test data.

- The linear column is assigned the forecasted values from the linear regression model (linear_forecast[1:n_test]).
- The random_Forest column is assigned the forecasted values from the Random Forest model (rf_forecast_Values[1:n_test]).
- The svr column is assigned the forecasted values from the SVR model (svr_forecast_values[1:n_test]).

By executing this code, you can create a dataframe (all_forecast) that combines the forecasts from the ARIMA model, linear regression model, random forest model, and svr model, by with the resultant actual values. This dataframe allows you to compare and analyse the forecasted values from different models in a unique manner.

Plotting All Models Together

To plot all models together, we have to install ***ggplot2*** package for the visual, below is the code for the ggplot graph for all the models we have used for analysis.

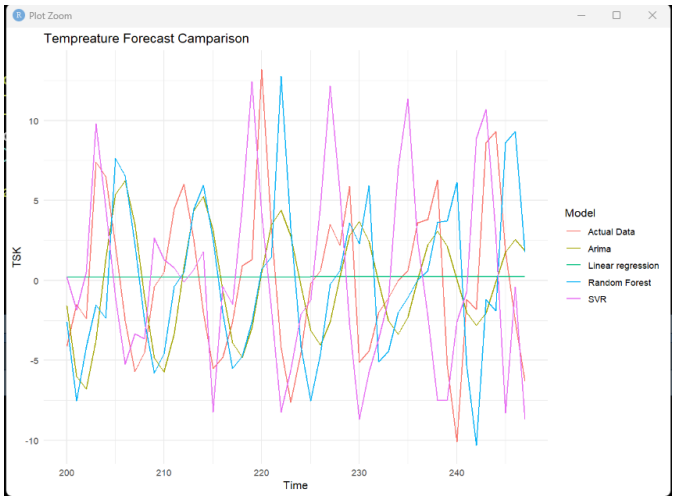


***Fig*.11.1: graph of all the models we used.**

Comparing all Models

While comparing all model we will use RMSE values, as discussed above, lesser value for the RMSE (Root mean square error) better is the accuracy, that means RMSE is the inversely proportional to the accuracy of the model.

Model	RMSE
ARIMA Model	3.026885
Linear Regression	4.8604
Random Forest Model	0.09867
Support Vector Model	5.32465

As per the above table, we can clearly see that we have Random value is the best model this analysis because it contains lowest value of RMSE. We also can se that with code in R studio as below in image ***fig.12***

```
# new data frame for RMSE values

rmse_values <- data.frame(
  model = c("ARIMA", "lINEAR REGRESSION", "RANDOM FOREST", "SVR"),
  rmse = c(arima_rmse, linear_rmse, rf_rmse, svr_rmse)
)

print(rmse_values)

best_model <- rmse_values[which.min(rmse_values$rmse), "model"]
cat("Best Model: ", best_model)
```

Fig.12: RMSE values new data-frame for each model

```
> best_model <- rmse_values[which.min(rmse_values$rmse), "model"]
> cat("Best Model: ", best_model)
Best Model:  RANDOM FOREST
>
```

Calculation best model which has the lowest number of RMSE value ie: Random Forest Model.

Data Visualisation

Data Visualisation in R refers to the creation of graphical representation which contain some insightful meaning and patterns in a dataset for the easy understanding of the dataset. R contain some powerful and wide range of packages which is designed for the data visualization such as ggplot2, base r graphics and plotty.

Data Visualization plays a very important role for the exploratory of data analysis as it helps data analyst for the data visualization and examine data patterns and trend within the datasets. It helps a lot in gaining a better understanding of the dataset by which we can identify the outliers and some other errors in a very clear manner.

In R, data visualization contains various types of charts including scatter plot, line graph, bar plots histogram, and more. This visualization can be modified with different colours add labels to enhance the clarity of the data set. R packages like ggplot2 can provide a very good approach for data visualization to create a complex data with easy visualization to understand the basic needs for the businesses and help in decision making for the businessperson. This approach allows for greater flexibility and customization of the plots using these packages.

This report we have used random forest model for data visualization as it contains lowest value of RMSE that means root mean square equation so we will use random forest model as for our data visualization and further process which will help different sectors with different problems as covered in our problem statement at the region of Barton turf which is located in the East of England.

Organisation Help

The result of accurate monitoring and measuring surface temperature in the Barton King would be very useful for several organization including public health organisation, agencies of government, Management of environment organisation. n a few public sector fields, Random Forest models could be used to forecast service want. These models, for illustrate, can approximate patient professions in the healthcare industry, allowing hospitals to schedule staffing levels and resource allocation appropriately. Below are some of the ways how the result will help for these organisations to make a better decision for public health-

- Public Health Organisation: Public health organisations can use the results which is predicted by the random forest model which help PHO to monitor surface temperature and patterns of the TSK. This information can be used to develop at target health involvement strategies and to identify the possible health impacts associated with temperature changes.
- Government Agencies: Government agencies are responsible for environmental and public safety which can be use the results of this random forest model prediction for the to inform and plan the management decision for health care and safety. For e.g. If there is a surface temperature shows a pattern for the unusually high temperature in certain areas in Barton turf, the government can issue as public notice for warning of high temperatures for a week and other measure which can help public with health in affected areas.
- Environmental Management Organisations: Environmental management organisations can be using that information to know about environmental

influences that can cause a skin problem or can cause an environmental problem such as humidity and rise in temperature. This information can be used to make a strategical environment manageable to make sure the areas of improving in planning of environment or environmental decision-making.

- Private Sector: This information can be used by private businessmen or private industries for their businesses for an example if they can predict about upcoming temperature of that area, they might increase the prices of air conditioners or any of their products which can be beneficial for them to increase the profit margin of their company. Using this information, they also can increase the production of their products if they get pre information about the temperature which might be increased in future. In Risk evaluation, Random Forest models may be used to evaluate hazards across a variety of disciplines. They can be used, for instance, in finance to analyse credit risk or evaluate investment possibilities. These models can be used to assess possible security risks and vulnerabilities in cybersecurity Systems that offer personalised goods or services to clients based on their interests, actions, or past data can be powered by Random Forest models. This can improve customer service, boost client interest, and boost revenue.

References

Chambers, J. M. (1992). Linear models. In J. M. Chambers & T. J. Hastie (Eds.), Statistical models in S (pp. 95-144).

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2015). Time series analysis: forecasting and control (4th ed.). Wiley. (pp. 60-199)

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to linear regression analysis (5th ed.). Wiley. (pp. 50-142)

Vapnik, V. N. (1995). The nature of statistical learning theory. Springer. (pp.80-100)

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer. (pp. 95-114).

Breiman, L. (2001). Random forests. Machine Learning, 45(1), (pp. 5-32).

Wickham, H. (2016). ggplot2: Elegant graphics for data analysis (2nd ed.). Springer.

Cleveland, W. S. (1993). Visualizing data. Hobart Press. (pp. 91-104).

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice (2nd ed.). OTexts. (pp. 75-149).

R Core Team. (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://www.R-project.org/

Helsel, D. R., & Hirsch, R. M. (2002). Statistical methods in water resources (3rd ed.). US Geological Survey. (pp. 95-111).

Hastie, T., Tibshirani, R., & Wainwright, M. (2015). Statistical learning with sparsity: The lasso and generalizations. CRC Press. (pp. 95-144).

Cleveland, W. S., & Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. Journal of the American Statistical Association, 83(403), (pp. 596-610).

Peng, R. D. (2018). R for environmental data science. CRC Press. (pp.693-773).

Lumley, T. (2010). Complex surveys: A guide to analysis using R. John Wiley & Sons. (pp. 130-188)

Verbyla, A. P., & Cressie, N. (2013). Spatial statistics and geostatistics: Theory and applications for geographic information science and technology. SAGE Publications (pp. 170-290)

Cowpertwait, P. S. P., & Metcalfe, A. V. (2009). Introductory time series with R. Springer. (pp. 450-510)

Banerjee, S., Carlin, B. P., & Gelfand, A. E. (2014). Hierarchical modeling and analysis for spatial data (2nd ed.). CRC Press. (pp. 43-73).

Tufte, E. R. (2001). The visual display of quantitative information (2nd ed.). Graphics Press.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian data analysis (3rd ed.). CRC Press. (pp. 92-156).

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice (2nd ed.). OTexts. (pp. 95-144).

Chen, C., & Liu, L. (2018). Time series forecasting with deep learning: A systematic literature review. Journal of Systems Science and Complexity, 31(2), (pp. 431-455).

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer Science & Business Media. pp. 13-73).

Wang, F., & Yao, Q. (2019). High-dimensional time series forecasting with neural networks: A review. International Journal of Forecasting, 35(4), (pp. 1438-1460.)

Borchers, H. W. (2015). Time series analysis in environmental science and policy. Wiley. pp. 313-739).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction (2nd ed.). Springer. (pp, 130-240)

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. J. (1990). STL: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics, 6(1), (pp. 3-73).

Wei, W. W. (2006). Time series analysis: Univariate and multivariate methods (2nd ed.). Pearson. pp. 30-77).

Hsu, C. W., Chang, C. C., & Lin, C. J. (2010). A practical guide to support vector classification. Department of Computer Science, National Taiwan University. pp. 30-73).

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, (pp. 159-175).

Word Count: 6376