Configuration Manual

MSc Research Project
Data Analytics

Garima Gupta
Student ID: X18182160

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Garima Gupta		
Student ID:	X18182160		
Programme:	Data Analytics	**Year:**	2019-2020
Module:	MSc Research Project		
Supervisor:	Dr. Catherina Mulwa		
Submission Due Date:	17/08/2020		
Project Title:	Mid Term Forecasting of Solar Power Generation in India: A Statistical Approach – Configuration Manual		
Word Count:	…………………… **Page Count 20**		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Garima Gupta

Date: 17/08/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Garima Gupta
x18182160

1 Introduction

This report will give a detailed explanation of all the steps taken to successfully complete this project. All the hardware and software requirements will also be discussed in to get the same results at every run.

2 System Configuration

2.1 Hardware Requirements

This project was implemented on Windows 10 machine with 8GB RAM and 512 GB of hard disk with the Ryzen 7 processor. Hardware specification is shown below in Figure 1:

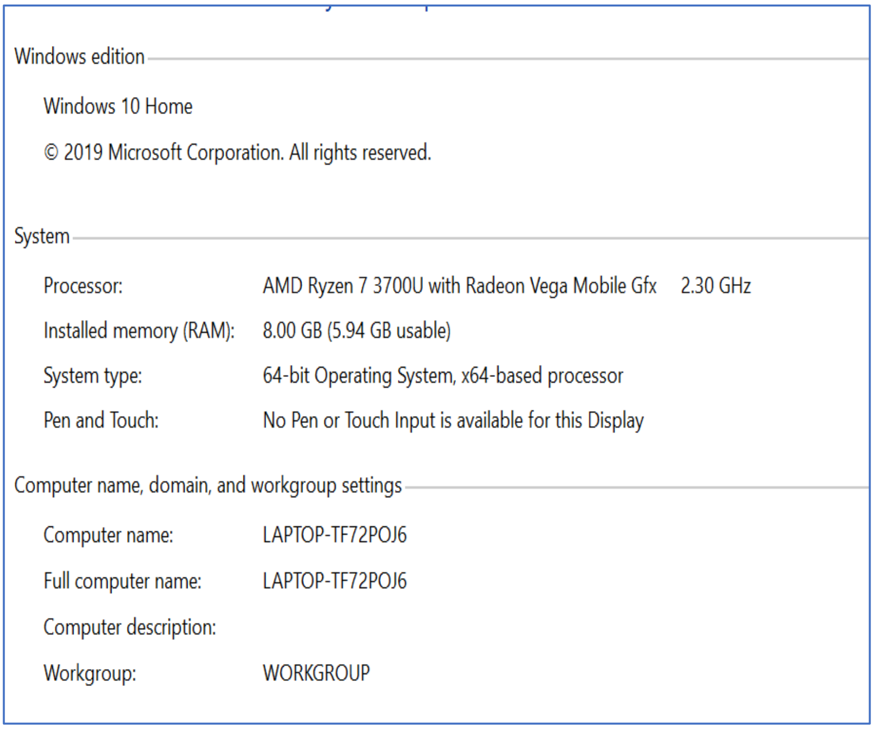

Figure 1: Hardware Requirements

2.2 Software Specification

Below mentioned software with specified versions were used for executing the project.

- Microsoft Excel 2019 was used for initial pre-processing of data which helped in saving time required for coding. The unnecessary columns and rows were dropped from the dataset and initial data check such as correct data types was checked at this stage using Excel.

- Python 3 Jupyter Notebook (Anaconda 3) was used for exploratory data analysis (EDA), such as scatterplot, heatmap. Jupyter notebook provides a lot of libraries that are very effective for data analysis and manipulation such as pandas, NumPy. For EDA visualizations matplotlib and seaborn libraries were used.
- Conversion of preprocessed data into time-series format was done using R programming language in Rstudio Integrated Development Environment. Different machine learning models were applied to time-series datasets using R. This programming language facilitates various time series and forecasting packages, which was the main reason for choosing R as development software for this project. The time-series plots and other forecast results were plotted using R.
- Tableau 2020.2 was used as a visualization tool for presenting the forecast results of the best-chosen model to help government authorities in making decisions for solar plant installation.

In R studio few packages were installed data cleaning, preprocessing, and applying forecasting models. A summary of addon packages is shown below in Figure 2:

```
#Installation of required packages
install.packages("forecast")
install.packages("chron")
install.packages("tseries")
install.packages("lubridate")
install.packages("zoo")
install.packages("lattice")
install.packages("ggplot2")
install.packages('prophet', type="source")
install.packages("nnfor")
install.packages("Metrics")
install.packages("CombMSC")
install.packages("ggpubr")
install.packages("TTR")
install.packages("Metrics")
install.packages("tsbox")
```

Figure 2: Required Add-on R packages

3 Project Development

To develop this research project, all tools specified above were used in data cleaning, preprocessing, and data modelling.

3.1 Data preparation

The raw data of Andhra Pradesh and Rajasthan was taken from the Central Pollution control board[1], initially, datasets were having 26304 rows and 5 columns. To achieve the goal of univariate time series analysis of solar irradiance forecasting project, Exploratory data analysis was performed on datasets using python 3. The raw data of both states can be seen below in Figure 3 and 4:

[1] 1https://app.cpcbccr.com/ccr/#/caaqm-dashboard-all/caaqm-landing/data/%7B%22state%22:%22Rajasthan%22,%22city%22:%22Jaipur%22,%22station%22:%22site_134%22%7D

CENTRAL POLLUTION CONTROL BOARD					
CONTINUOUS AMBIENT AIR QUALITY					
					Date: Wednesday, Aug 05 2020
					Time: 08:34:08 AM
Parameter	SR,BP,RH,WS				
AvgPeriod	24 Hours				
From	14-06-2017T00:00:00Z 00:00				
To	14-06-2020T00:00:59Z 00:00				
Police Commissionerate, Jaipur - RSPCB					
Prescribed Standards		NA	NA	NA	NA
Exceeding Standards		NA	NA	NA	NA
Remarks					
From Date	To Date	SR	BP	RH	WS
14-06-2017 00:00	15-06-2017 00:00	119.12	767.36	34.67	0.89
15-06-2017 00:00	16-06-2017 00:00	94.87	767.22	45.67	0.8
16-06-2017 00:00	17-06-2017 00:00	94.79	767.24	38.44	1.11
17-06-2017 00:00	18-06-2017 00:00	91.82	767.2	40.59	0.96
18-06-2017 00:00	19-06-2017 00:00	89.85	767.17	41.78	0.8
19-06-2017 00:00	20-06-2017 00:00	100.27	767.13	40.76	0.78
20-06-2017 00:00	21-06-2017 00:00	95.66	767	42.73	1
21-06-2017 00:00	22-06-2017 00:00	105.41	767.05	42.88	0.96
22-06-2017 00:00	23-06-2017 00:00	88.02	766.94	51.25	1.01
23-06-2017 00:00	24-06-2017 00:00	84.22	766.9	50.01	0.94
24-06-2017 00:00	25-06-2017 00:00	None	794.25	50.96	0.94
25-06-2017 00:00	26-06-2017 00:00	None	823.77	45.12	0.86
26-06-2017 00:00	27-06-2017 00:00	None	833.86	47.66	1
27-06-2017 00:00	28-06-2017 00:00	None	820.02	63.57	1.07
28-06-2017 00:00	29-06-2017 00:00	1586.84	777.79	87.2	0.93
29-06-2017 00:00	30-06-2017 00:00	65.61	766.7	92.17	1.06

Figure 3: Rajasthan's Raw data

CENTRAL POLLUTION CONTROL BOARD					
CONTINUOUS AMBIENT AIR QUALITY					
					Date: Friday, Aug 07 2020
					Time: 02:59:40 AM
State	Andhra Pradesh				
City	Visakhapatnam				
Station	GVM Corporation, Visakhapatnam - APPCB				
Parameter	SR,BP,RF,WS				
AvgPeriod	24 Hours				
From	14-06-2017T00:00:00Z 00:00				
To	14-06-2020T00:00:59Z 00:00				
GVM Corporation, Visakhapatnam - APPCB					
Prescribed Standards		NA	NA	NA	NA
Exceeding Standards		NA	NA	NA	NA
Remarks					
From Date	To Date	SR	BP	RF	WS
14-06-2017 00:00	15-06-2017 00:00	18.62	742.05	0.06	1.49
15-06-2017 00:00	16-06-2017 00:00	72.64	741.07	0.01	2.41
16-06-2017 00:00	17-06-2017 00:00	76.24	738.83	0.01	2.35
17-06-2017 00:00	18-06-2017 00:00	184.81	736.9	0	3.01
18-06-2017 00:00	19-06-2017 00:00	179.7	734.4	0	2.95
19-06-2017 00:00	20-06-2017 00:00	119.42	731.84	0	2.25
20-06-2017 00:00	21-06-2017 00:00	169.01	731.65	0	2.5
21-06-2017 00:00	22-06-2017 00:00	144.11	733.83	0	2.03
22-06-2017 00:00	23-06-2017 00:00	193.22	731.13	0	2.12
23-06-2017 00:00	24-06-2017 00:00	180.17	732.03	0.13	1.53
24-06-2017 00:00	25-06-2017 00:00	123.8	729.5	0.07	1.32
25-06-2017 00:00	26-06-2017 00:00	36.33	729.91	0.02	0.96
26-06-2017 00:00	27-06-2017 00:00	71.1	730.75	0	2.92

Figure 4: Andhra Pradesh Raw data

From Date is of no use in the analysis of the project, so it was dropped from the datasets, and data was imported to Jupyter notebook for exploratory data analysis. Figure 5 and 6 shows the first 5 rows of imported datasets:

```python
# Importing Rajasthan data
import pandas as pd
RJ_df = pd.read_excel('Rajasthan_jaipur_daily.xlsx')
RJ_df.head()
```

	To Date	SR	BP	RH	WS
0	15-06-2017 00:00	119.12	767.36	34.67	0.89
1	16-06-2017 00:00	94.87	767.22	45.67	0.8
2	17-06-2017 00:00	94.79	767.24	38.44	1.11
3	18-06-2017 00:00	91.82	767.2	40.59	0.96
4	19-06-2017 00:00	89.85	767.17	41.78	0.8

Figure 5: First 5 rows of Rajasthan data

```python
# Importing Andhra Pradesh data
AP_df = pd.read_excel('AP_Vizag_daily.xlsx')
AP_df.head()
```

	To Date	SR	BP	RH	WS
0	15-06-2017 00:00	18.62	742.05	0.06	1.49
1	16-06-2017 00:00	72.64	741.07	0.01	2.41
2	17-06-2017 00:00	76.24	738.83	0.01	2.35
3	18-06-2017 00:00	184.81	736.9	0	3.01
4	19-06-2017 00:00	179.7	734.4	0	2.95

Figure 6: First 5 rows of Andhra Pradesh's data

To find the relationship between the columns, the scatterplot of each variable with solar irradiance was drawn. The scatterplots along with code to draw it are shown in Figure 6, 7, 8 for Rajasthan's data and 9, 10, 11 for Andhra Pradesh's data.

```python
In [7]: #Drawing Scatterplot
        import matplotlib.pyplot as plt
        plt.plot(RJ_df['SR'],RJ_df['BP'], marker = '.', linestyle = 'None')
        plt.xlabel("Solar irradiance")
        plt.ylabel("Bar pressue")

Out[7]: Text(0, 0.5, 'Bar pressue')
```

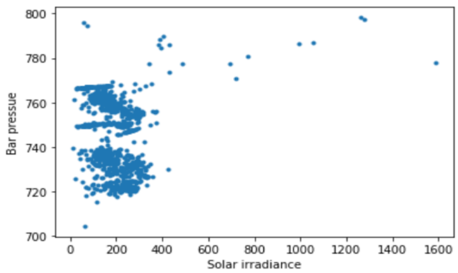

Figure 7: Scatter plot of Rajasthan's Bar pressure

```
In [8]: plt.plot(RJ_df['SR'],RJ_df['RH'], marker = '.', linestyle = 'None')
        plt.xlabel("Solar irradiance")
        plt.ylabel("Relative humidity")

Out[8]: Text(0, 0.5, 'Relative humidity')
```

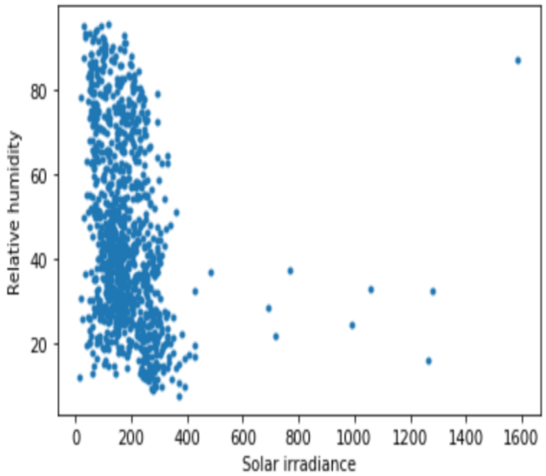

Figure 8: Scatter plot of Rajasthan's Humidity

```
 [9]: plt.plot(RJ_df['SR'],RJ_df['WS'], marker = '.', linestyle = 'None')
      plt.xlabel("Solar irradiance")
      plt.ylabel("Wind Speed")

t[9]: Text(0, 0.5, 'Wind Speed')
```

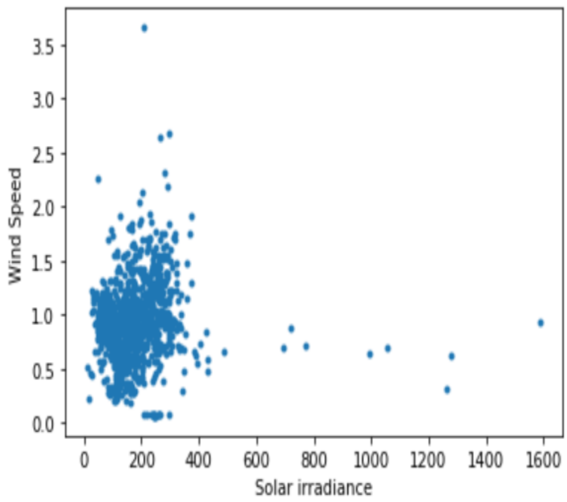

Figure 9: Scatter plot of Rajasthan's Wind speed

```
#Drawing Scatterplot
plt.plot(AP_df['SR'],AP_df['BP'], marker = '.', linestyle = 'None')
plt.xlabel("Solar irradiance")
plt.ylabel("Bar pressue")
```

```
Text(0, 0.5, 'Bar pressue')
```

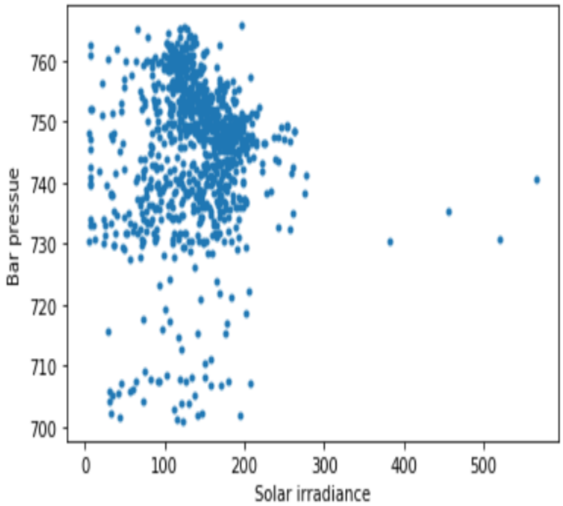

Figure 10: Scatter plot of Andhra Pradesh's Bar pressure

```
plt.plot(AP_df['SR'],AP_df['RH'], marker = '.', linestyle = 'None')
plt.xlabel("Solar irradiance")
plt.ylabel("Relative humidity")
```

```
Text(0, 0.5, 'Relative humidity')
```

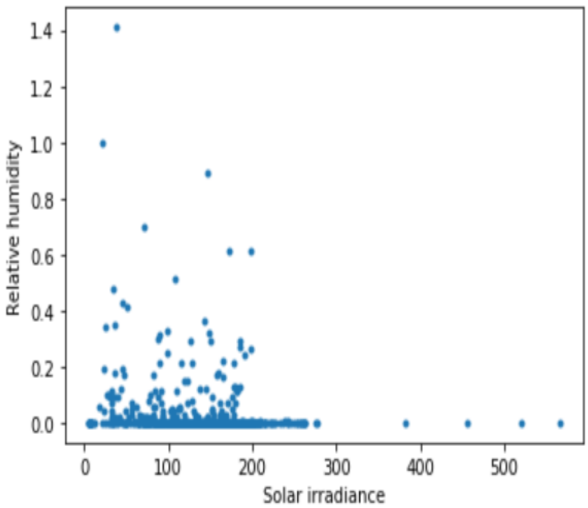

Figure 11: Scatter plot of Andhra Pradesh's Humidity

```
plt.plot(AP_df['SR'],AP_df['WS'], marker = '.', linestyle = 'None')
plt.xlabel("Solar irradiance")
plt.ylabel("wind Speed")
```

```
Text(0, 0.5, 'wind Speed')
```

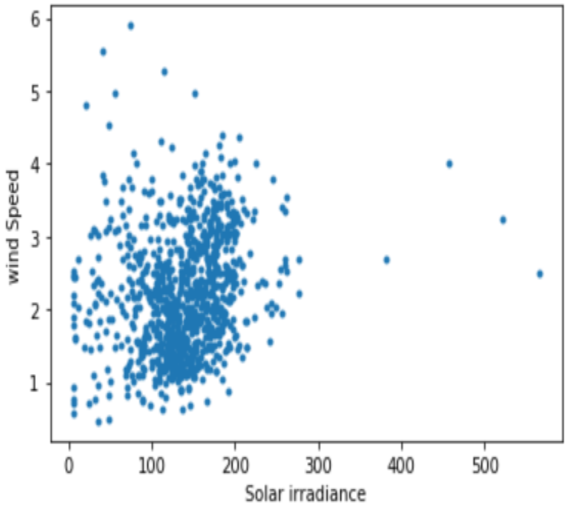

Figure 12: Scatter plot of Andhra Pradesh's wind speed

Figure 13 and 14 has been drawn to check the correlation between the individual parameters and it was found that there is a weak relationship between parameters, hence solar irradiance was only considered for analysis.

```
#Plotting Heatmap
import seaborn as sns
sns.heatmap(corr, annot =True, cmap="Blues")
plt.show()
```

Figure 13: Heatmap of Rajasthan data

```python
#Plotting Heatmap
import seaborn as sns
sns.heatmap(corr1, annot =True, cmap="YlGnBu")
plt.show()
```

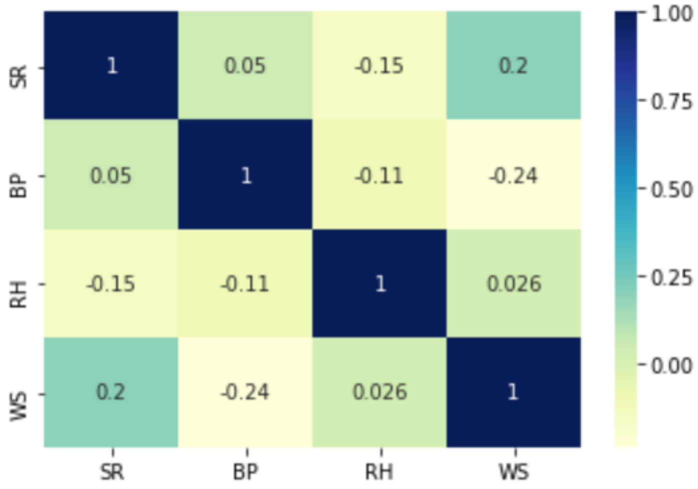

Figure 14: Heatmap of Andhra Pradesh's data

After EDA data was imported to Rstudio for data cleaning and preprocessing of converting the dataset into time-series data. Figure 15 shows the code for cleaning and preprocessing of datasets:

```r
setwd("C:/Users/garim/Desktop/NCI/Semester 3/Research Project")

#############Rajasthan Analysis#######################

Rajasthan <- read.xlsx("Rajasthan_Jaipur_Daily_SR.xlsx",1)
summary(Rajasthan)
str(Rajasthan)

#Feature Engineering to convert To.Date column (containing timestamp) into Date column

Rajasthan = separate(Rajasthan, col = To.Date, into = c("Date"), sep = 10, remove=T)
Rajasthan$Date=strptime(Rajasthan$Date, format= "%d-%m-%Y")
Rajasthan$SR = as.numeric(as.character(Rajasthan$SR))
str(Rajasthan)
summary(Rajasthan)

#Defining zoo variable for creating time series

temp= zoo(Rajasthan %>% select(2), order.by = Rajasthan$Date)
temp$SR <- tsclean(temp$SR)

Rajasthan.ts <- ts(temp, start= c(2017, 06, 15),  frequency=365)

autoplot(Rajasthan.ts) +
  ggtitle("Solar irradiance data of Rajasthan(2017-2020)") +
  xlab("Year") +
  ylab("Solar irradiance (W/m2)")
```

Figure 15: Cleaning and preprocessing of Rajasthan's dataset

Plots of time series dataset of both states were plotted with or without seasonality as shown in Figure 16 and 17.

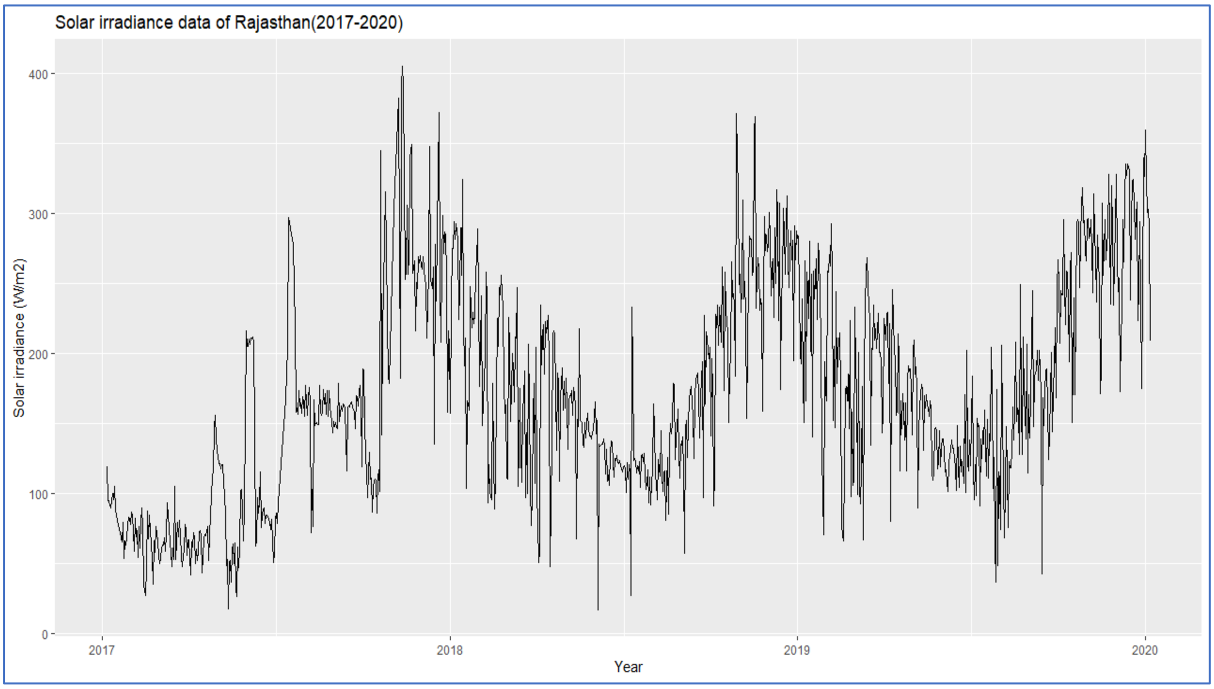

Figure 16: Rajasthan' s time-series data

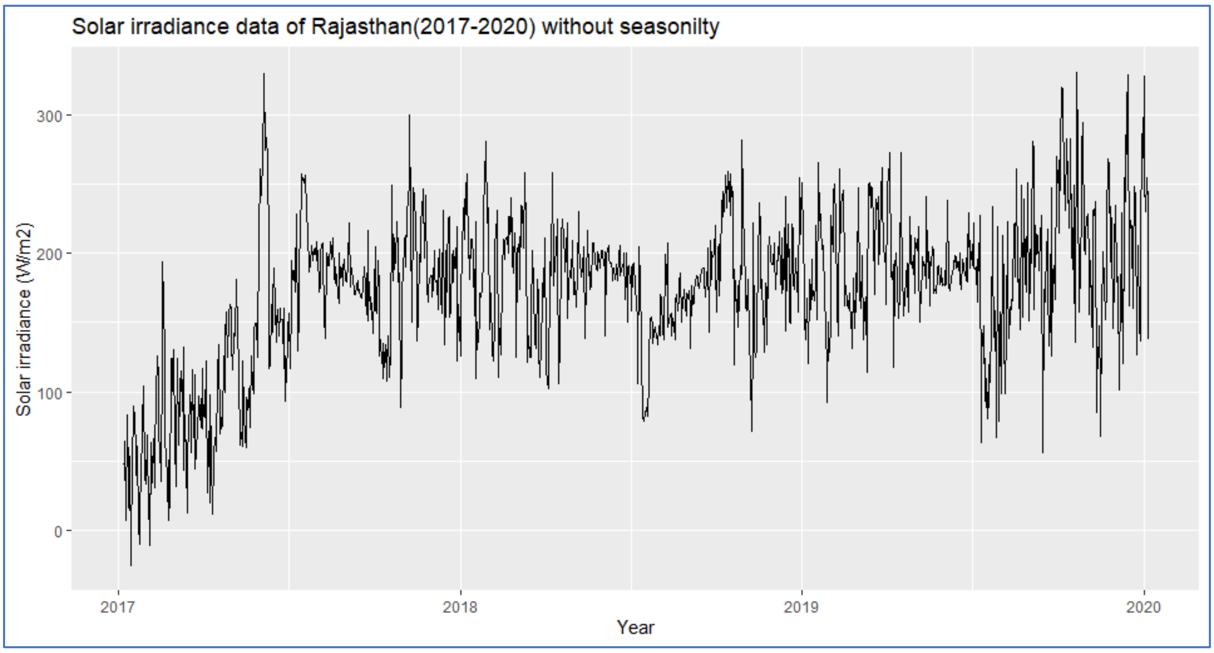

Figure 17: Rajasthan's time-series data without seasonality

From the comparison of both figures, it can be observed that Figure 16 was seasonal data but 17 is non-seasonal. Figure 18 shows the code of different tests performed on time series data to make it ready for modelling.

```
#Dickey-Fuller test for Stationarity check
adf.test(RJadj)

#Ljung-Box Test for white noise

Box.test(RJadj, lag = 24, fitdf = 0, type = "Ljung")

#Checking normality using Wilk shapiro test
shapiro.test(RJadj)

# Q-Q plot for checking data linearity
ggqqplot(temp$SR)+
  ggtitle("Rajasthan Solar irradiance Data Linearity")

#spliting the original data for training and testing dataset with a ratio of 75-25 -> Se

Rajasthan1 <-splitTrainTest(RJadj, numTrain = length(RJadj) - 274)

#spliting the original data for training and testing dataset with a ratio of 80-20 -> Se
Rajasthan2 <-splitTrainTest(RJadj, numTrain = length(RJadj) - 220)
```

Figure 18: Data Analysis for modelling

4 Model Implementation and results

This section discusses the implementation of selected models and their results. As the code of both datasets is the same, so details of Rajasthan state analysis has been shown here.

4.1 ARIMA Model:

Figure 19, 20, and 21 shows the code for model implementation, residual plot and forecast results of ARIMA model respectively.

```
####Using Training Set 1 ################

RJarima1 <-auto.arima(Rajasthan1$train, lambda= NULL, stationary = T,
                      stepwise=F, trace=F, approximation=F, biasadj=F)
checkresiduals(RJarima1)


#By looking at the residuals, it can be seen that it is white noise, so the model is
RJarima1.forecast <-forecast(RJarima1,h = 274)

autoplot(RJarima1.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")

RJarima1.acc<-accuracy(RJarima1.forecast)

####Using Training Set 2 ################

RJarima2 <-auto.arima(Rajasthan2$train,lambda=NULL, stationary = T,
                      stepwise=F, trace=F, approximation=F, biasadj=F)
checkresiduals(RJarima2)

#By looking at the residuals, it can be seen that it is white noise, so the model is
RJarima2.forecast <-forecast(RJarima2,h=220)

autoplot(RJarima2.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")

RJarima2.acc<-accuracy(RJarima2.forecast)
```

Figure 19: Code for ARIMA model

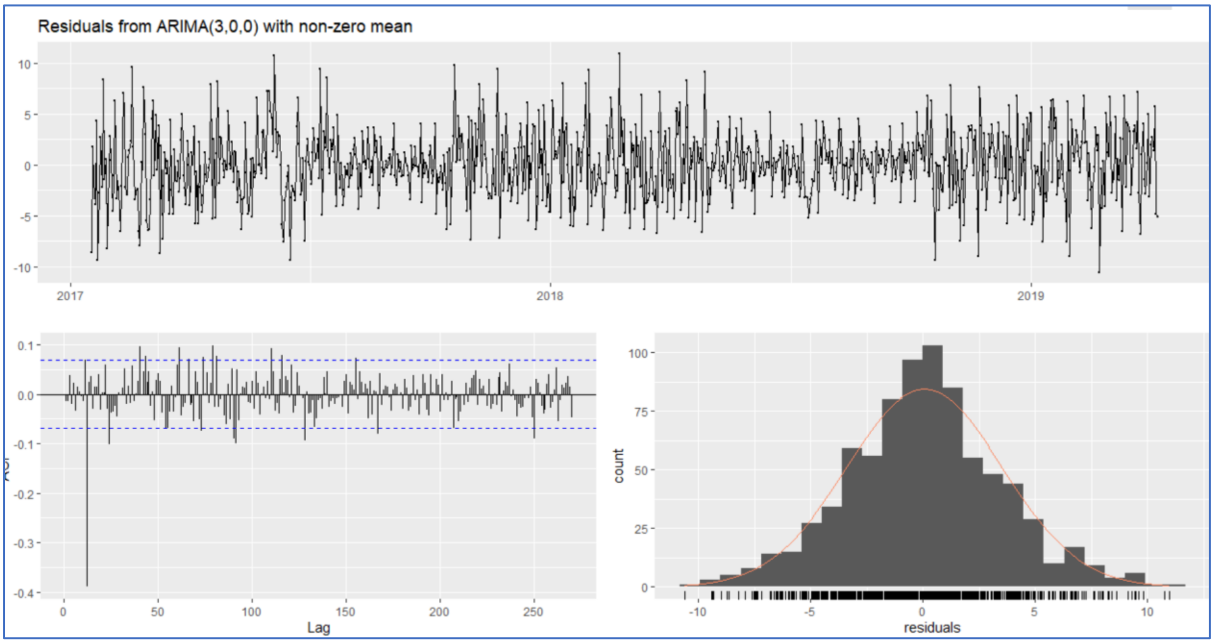

Figure 20: Residual plot of ARIMA Model

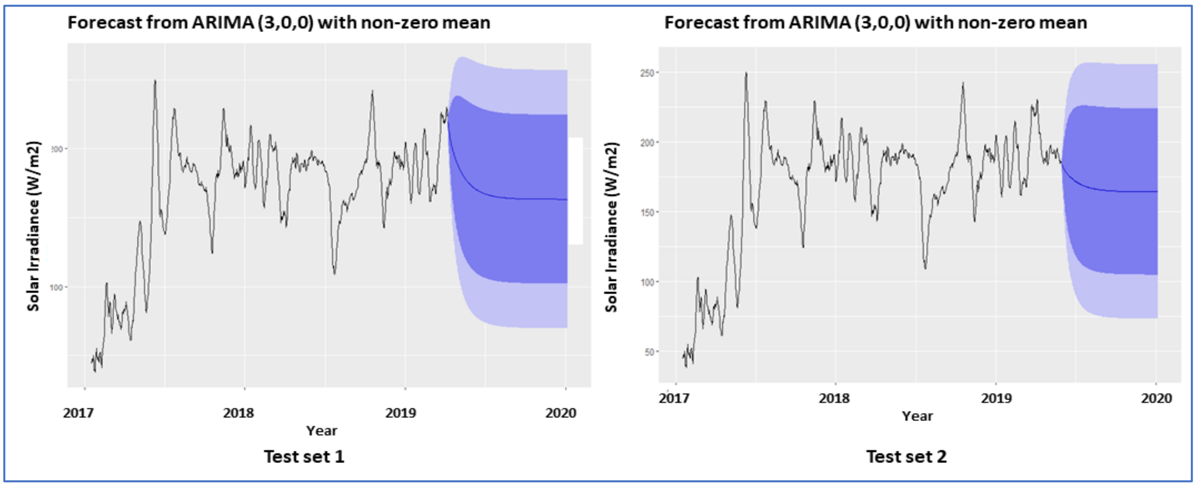

Figure 21: ARIMA forecast Plot

4.2 Simple Exponential Smoothing Model

Figure 22 and 23 shows the code for model implementation, and forecast results of Simple exponential smoothing model respectively.

```r
RJses1 <- ses(Rajasthan1$train, h = 274, lambda = NULL, initial="optimal", biasadj=F)

#Checking the errors to evaluate the model

RJses1.acc<-accuracy(RJses1)

# Add the one-step forecasts for the training data to the plot

autoplot(Rajasthan1$train) + autolayer(fitted(RJses1), series = "ses") +
  xlab("Year") +
  ggtitle("Simple Exponential Smoothing forecast of Rajasthan training set1") +
  ylab("Solar irradiance(W/m2)")

####Using Training Set 2 ###############

RJses2 <- ses(Rajasthan2$train, h = 220, lambda = NULL, initial="optimal", biasadj=F)

#Checking the error to evaluate the model

RJses2.acc<-accuracy(RJses2)

# Add the one-step forecasts for the training data to the plot

autoplot(Rajasthan2$train) + autolayer(fitted(RJses2), series = "ses") +
  xlab("Year") +
  ggtitle("Simple Exponential Smoothing forecast of Rajasthan training set2") +
  ylab("Solar irradiance(W/m2)")
```

Figure 22: R code for SES model

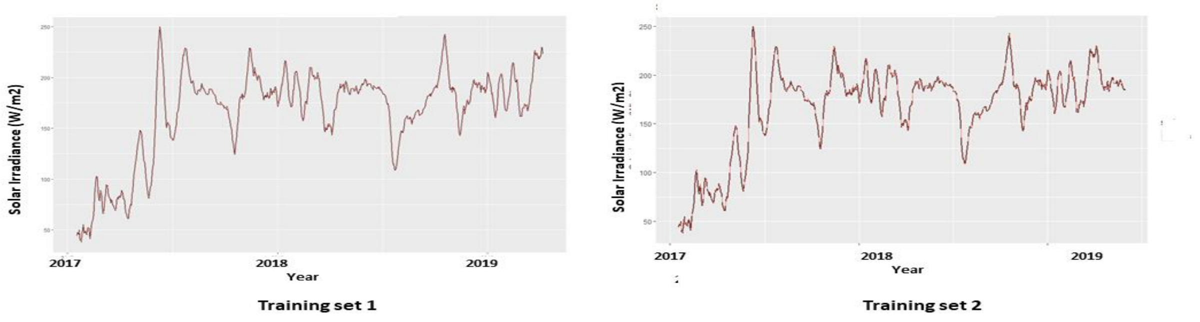

Figure 23: Forecast plots of SES

4.3 Dynamic Harmonic Regression Model

Figure 24 and 25 shows the code for model implementation, and forecast results of the Dynamic Harmonic regression model respectively. The Value of K was selected by trying different values and K =2 gave the least AICc value, so it was chosen.

```
####Using Training Set 1 ################

RJdhr1<- auto.arima(Rajasthan1$train, xreg= fourier(Rajasthan1$train, K=2),
                   lambda = 0, stationary = T,stepwise=F, trace=F, approximation=F, bia

#Checking for the minimum AICc value to determine the value of K
summary(RJdhr1)
checkresiduals(RJdhr1)

RJdhr1.forecast<-forecast(RJdhr1, xreg= fourier(Rajasthan1$train, K=2, h=274))
RJdhr1.acc<-accuracy(RJdhr1.forecast)

autoplot(RJdhr1.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")

####Using Training Set 2 ################

RJdhr2<- auto.arima(Rajasthan2$train, xreg= fourier(Rajasthan2$train, K=2),
                    lambda = 0, stationary = T,stepwise=F, trace=F, approximation=F, bia


#checking for the minimum AICc value to determine the value of K
summary(RJdhr2)
checkresiduals(RJdhr2)

RJdhr2.forecast<-forecast(RJdhr2, xreg= fourier(Rajasthan2$train, K=2, h=220))
```

Figure 24: R Code for DHR model

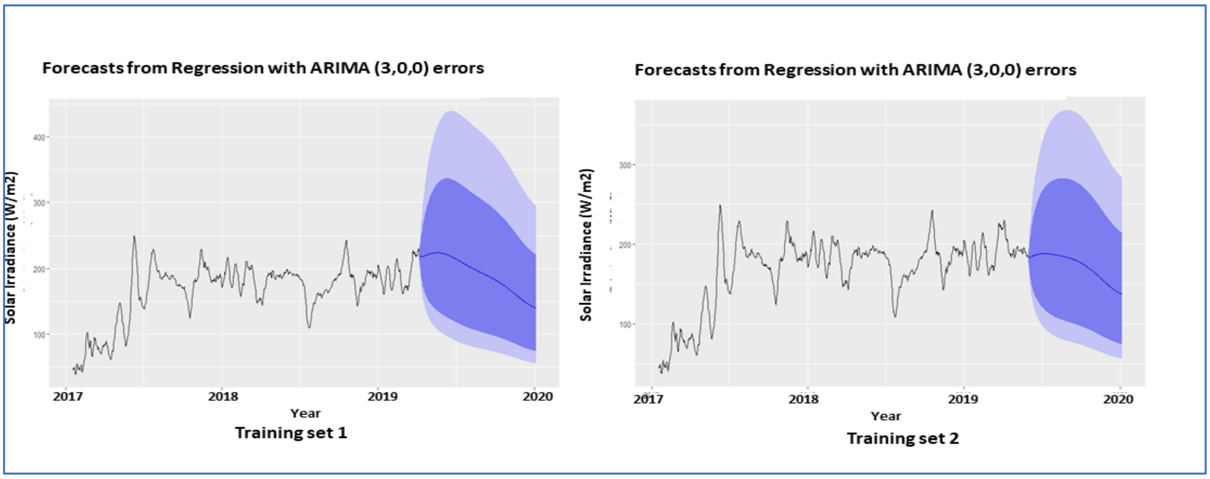

Figure 25: SES Forecast Plot

4.4 TBATS Model

Figure 26 and 27 shows the code for model implementation, and forecast results of TBATS model respectively.

```
#using training set 1 ################

RJtbats1 <- tbats(Rajasthan1$train, use.box.cox = NULL,
                  use.trend = NULL, biasadj = FALSE)

RJtbats1.forecast<- forecast(RJtbats1, h=274)

RJtbats1.acc<-accuracy(RJtbats1.forecast)

autoplot(RJtbats1.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")

#using training set 2 ################

RJtbats2 <- tbats(Rajasthan2$train, use.box.cox = NULL,
                  use.trend = NULL, biasadj = FALSE)

RJtbats2.forecast<- forecast(RJtbats2, h=220)

RJtbats2.acc<-accuracy(RJtbats2.forecast)

autoplot(RJtbats2.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")
```

Figure 26: R code for TBATS model

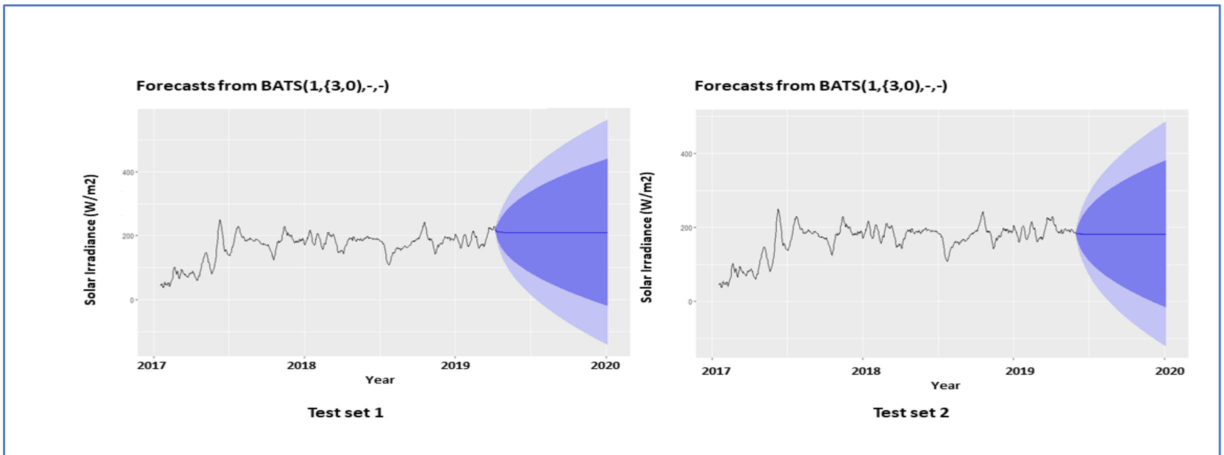

Figure 27: TBATS forecast plot

4.5 Neural Network Model

Figure 28 and 29 shows the code for model implementation, and forecast results of Neural Network respectively.

```r
#using training set 1 ################

RJnn1=nnetar(Rajasthan1$train,p=10,P=2,size =6,repeats=100,
             lambda = NULL, scale.inputs=F)

RJnn1.forecast <-forecast(RJnn1, h=274)

RJnn1.acc<-accuracy(RJnn1.forecast)

autoplot(RJnn1.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")

#using training set 2 ################

RJnn2=nnetar(Rajasthan2$train,p=10,P=2,size =6,repeats=100,
             lambda = NULL, scale.inputs=F)

RJnn2.forecast <-forecast(RJnn2, h=220)

RJnn2.acc<-accuracy(RJnn2.forecast)

autoplot(RJnn2.forecast) +
  xlab("Year") +
  ylab("Solar irradiance(W/m2)")
```

Figure 28: R code for Neural Network Model

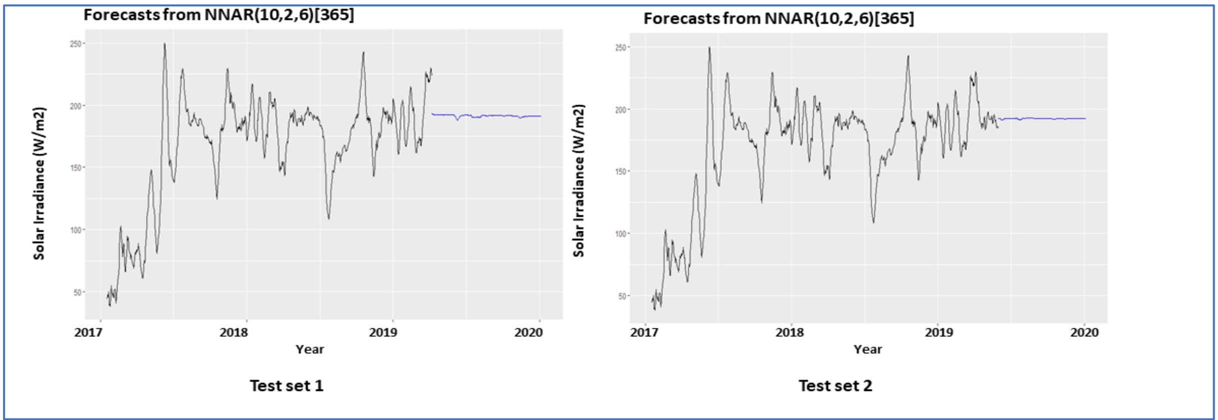

Figure 29: Forecast plot of Neural Network

4.6 Prophet Model

Figure 30 and 31 shows the code for model implementation, and forecast results of Prophet model respectively. An extra step is added in code for making a dataframe that is accepted by the prophet model for data modelling. Dataframe with 2 columns named as “ds” and “y” was created by converting time-series data back to dataframe.

```r
########################## Prophet Model #################################################

#Creating a dataframe with column named as ds(datestamp)and y(solar irradiance)
Rajasthan.df <- ts_df(RJadj)
Rajasthan.df <- separate(Rajasthan.df, col = time, into = c("ds"), sep = 10, remove=T)
Rajasthan.df <- rename(Rajasthan.df, y = value)

Rajasthan.temporal <- Rajasthan.df

####Using Training Set 1 ################

# Loading Metrics package for evaluation parameters

library(Metrics)

RJ.prophet1<-prophet(Rajasthan.temporal[1:811,], changepoints = NULL,
                     seasonality.mode = 'additive', daily.seasonality=F,fit = T)

#Creating test dataset
#RJfuture1 <- make_future_dataframe(RJ.prophet1, periods = 274, freq = "day")

RJfuture1 <- data.frame(ds = Rajasthan.temporal[812:841,1])
RJfuture1 <- separate(RJfuture1, col = ds, into = c("ds"), sep = 10, remove=T)
RJfuture1$ds=strptime(RJfuture1$ds, format= "%Y-%m-%d")


RJ.prophet1forecast <- predict(RJ.prophet1, RJfuture1, type="response")
dyplot.prophet(RJ.prophet1, RJ.prophet1forecast)
```

```r
####### Error calculation

RJactual1<- Rajasthan.temporal[812:841,2]

RJprophet1.acc<-data.frame(rmse=rmse(RJactual1, RJ.prophet1forecast$yhat),
                           mae=mae(RJactual1, RJ.prophet1forecast$yhat),
                           mape=mape(RJactual1, RJ.prophet1forecast$yhat))
detach("package:Metrics", unload = TRUE)
```

Figure 30: R code for Prophet Model

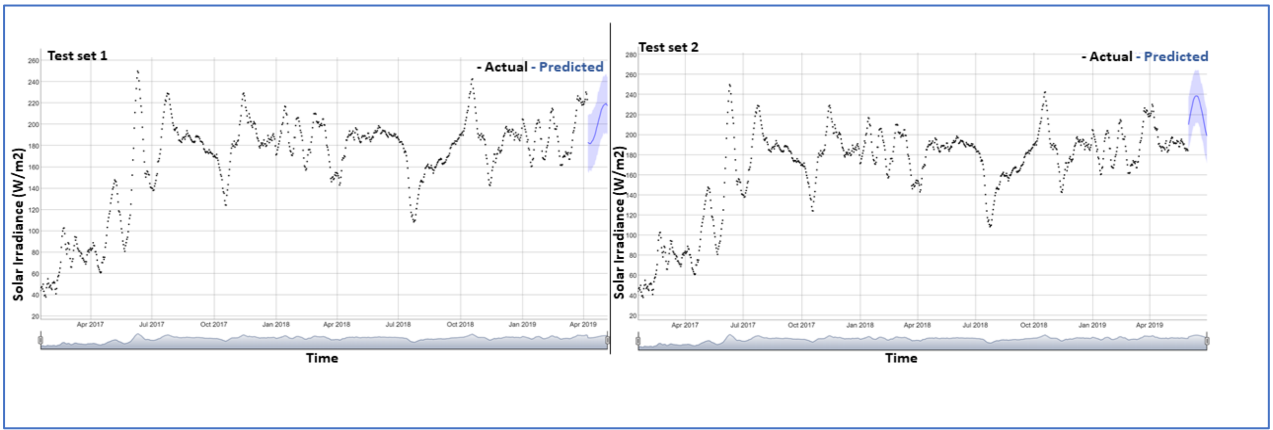

Figure 31: Forecast Result of Prophet Model

5 Comparison of Model's performance

Table 1 compares the performance of implemented models using RMSE, MAE, and MAPE as evaluation metrics.

Table 1: Comparison of the result of implemented Models

Applied Models	Rajasthan			Andhra Pradesh		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA	3.429134	2.64023	1.870284	1.92607	1.39542	1.049723
SES	4.346164	3.29734	2.26291	2.276351	1.642982	1.256257
DHR	3.443069	2.660631	1.861069	1.927582	1.396146	1.050017
TBATS	3.395737	2.625447	1.837209	1.978833	1.451111	1.097205
Neural	16.38944	12.70494	6.622763	8.938446	7.521864	5.851798
Prophet	21.23669	19.01291	0.097867	16.95289	13.98698	0.102526

6 Calculation of Solar Power

The solar power was calculated for both states using the ARIMA model for the next year, as ARIMA outperformed every model. R code for the calculation of solar energy generation is given in Figure 32. Figure 33 portrays the forecasted power of both states.

```
##############Forecasting solar irradiance of next one year###################
#As ARIMA performed best for forecasting on both datasets with 80-20 split training set
#so ARIMA will be used for forecasting irradiance of Rajasthan as well as
#Andhra Pradesh for next one year (365 days)

RJfit <-auto.arima(Rajasthan2$train,lambda=NULL, stationary = T,
                      stepwise=F, trace=F, approximation=F, biasadj=F)
RJSR.forecast <-forecast(RJfit,h=365)

#########################################

APfit <-auto.arima(Andhra2$train,lambda=NULL, stationary = T,
                   stepwise=F, trace=F, approximation=F, biasadj=F)
APSR.forecast <-forecast(APfit,h=365)


# Calculating Solar Power with default parameter values
# SP = A*r*H*PR
#SP = Solar Power (KWh)
#A = Solar Panel Area (m2) [average area of solar panel is 1.6 m2, so A will be 1.6]
#r = Solar panel Efficiency (%) [efficiency ranges between 10-15%,
#we will consider it as 15%, hence r = 0.15]
#H = Average solar irradiance(W/m2) [forecasted value]
#PR = performance ratio default 0.75

ForecastedSP<-data.frame("Rajasthan_Solar_power"=(1.6 * 0.15 * 0.75 * RJSR.forecast$mean),
                     "AndhraPradesh_Solar_Power"=(1.6 * 0.15 * 0.75 * APSR.forecast$mean))
write.csv(ForecastedSP,'Forecast_SR.csv')
```

Figure 32: R code for Solar energy calculation

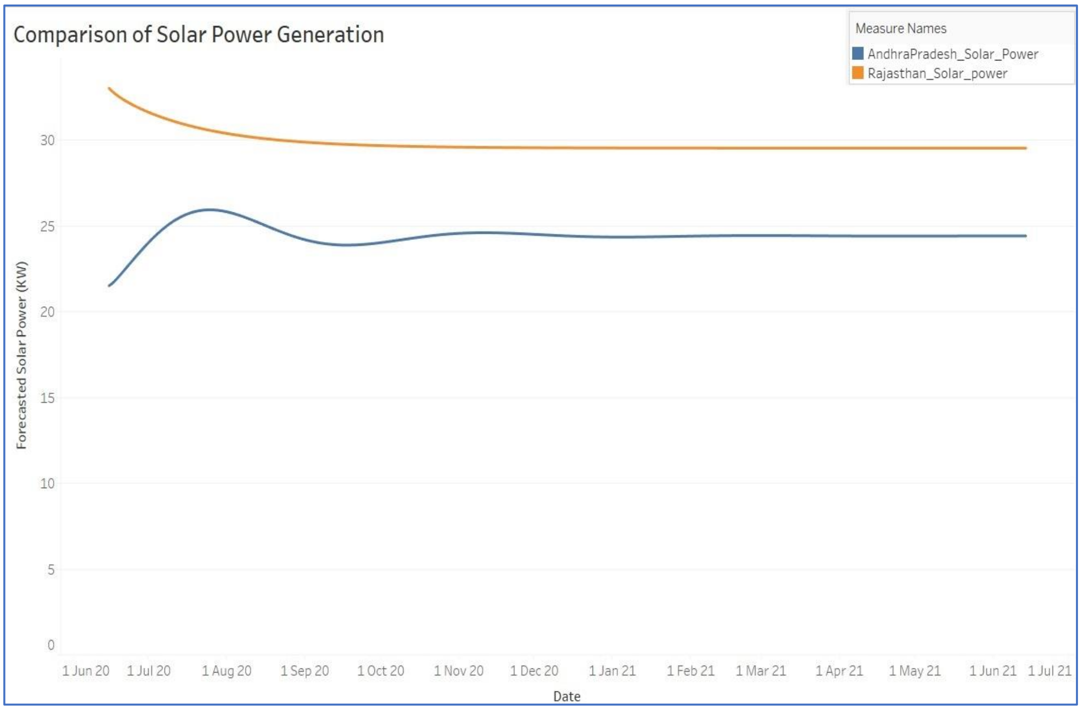

Figure 33: Forecasted Solar energy of Rajasthan & Andhra Pradesh