

Configuration Manual

MSc Research Project
Data Analytics

Yogesh Maruti Patil
Student ID: x17169828

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Yogesh Maruti Patil
Student ID:	x17169828
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	16/09/2019
Project Title:	Configuration Manual
Word Count:	671
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:	
Date:	15th September 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Yogesh Maruti Patil
x17169828

1 System Configuration

The systems configuration plays an important role in order to start up with a new project. It is always suggested to look for the specifications first and download the required software accordingly. The configuration of the system used in this project is as shown in figure 1.

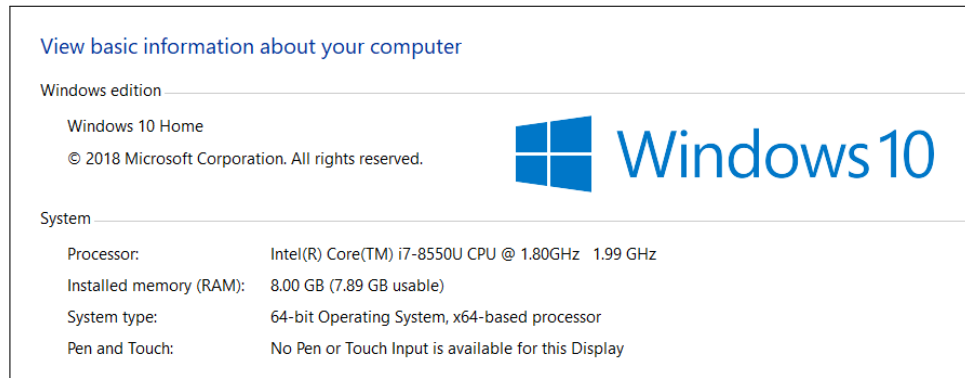


Figure 1: System Configuration

2 Software Installations

Some software installations were required to be done for this project. The softwares required for this project included R, R Studio, Tableau, and Microsoft Excel. The steps involved in their installation are being enlisted in this section.

2.1 Installation of R and R Studio

R is a software environment that comes with a Graphical User interface. R studio is an Integrated Development Environment. The steps involved in this installation are as follows:

1. The R installer for Windows can be downloaded from the CRAN website ¹ as shown in 2. The setup is quite easy, just press Next when required according to the instruction.

¹<https://cran.r-project.org/>

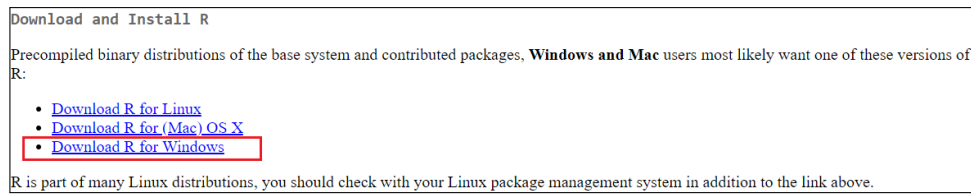


Figure 2: R installation from CRAN website

2. The second step involves installing the R studio. R studio installer can be downloaded from the RStudio website ² as shown in 3. Download it according to system specification (e.g.32/64 bits). The setup for installation are quite easy, just press Next when required according to the instruction.

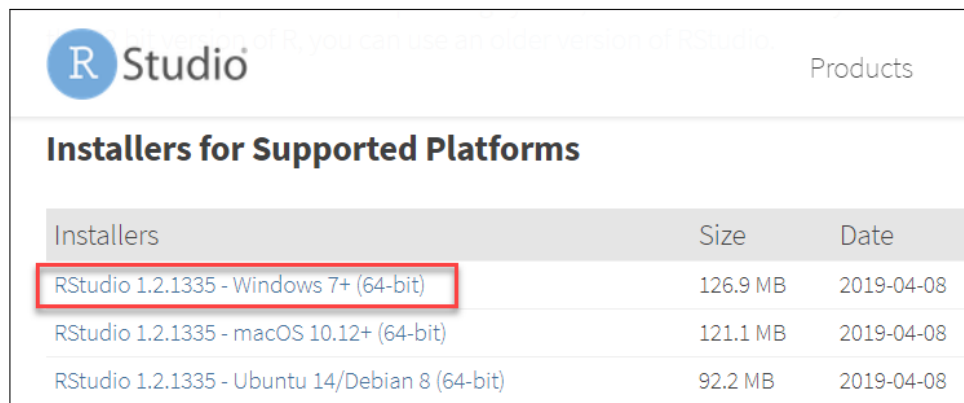


Figure 3: R Studio installation from RStudio website

3. Once the above two steps are done R Studio is ready to use. The initial window of R studio is as shown in 4. There after one can start coding.

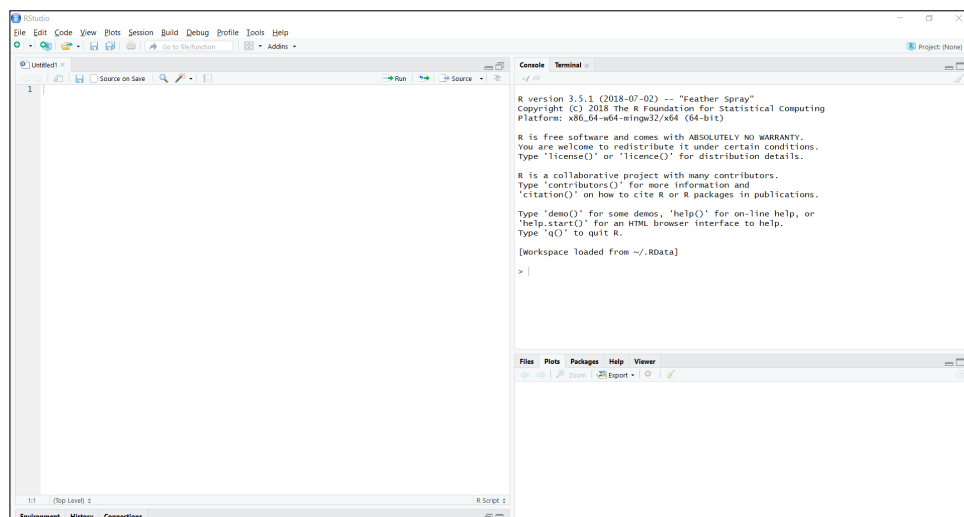


Figure 4: R Studio

²<https://www.rstudio.com/products/rstudio/download/download>

2.2 Installation of Tableau

Tableau is used for the visualization of the data. The steps involved in its installation are as follows:

1. The Tableau installer for Windows can be downloaded from the Tableau website ³ as shown in 5. The setup is quite easy, just press Next when required according to the instruction.

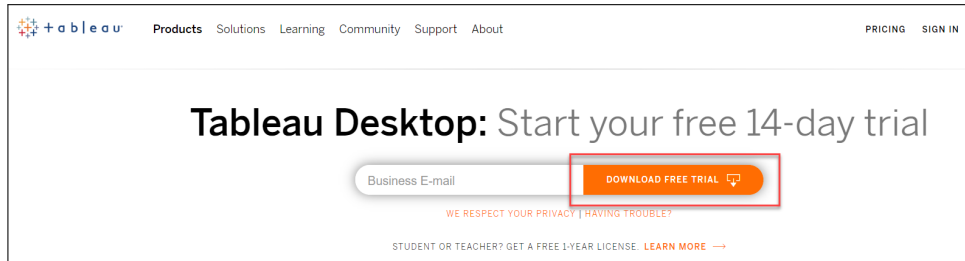


Figure 5: Tableau Website

2. Once the installation is done, one can connect any data file into Tableau and start working on charts.

3 Data Download

Data is downloaded from two sources for this project. The detailed information about them are as follows.

3.1 Solar Irradiance Data

For this project the solar data was downloaded from Solcast website ⁴. Here free solar irradiance data is available for academic use. The steps involved in downloading the data are as follows:

1. Go to the Solcast website and register a new account as a student as shown in figure 6 and 7

³<https://www.tableau.com/products/desktop/download?os=windows>

⁴<https://solcast.com.au/solar-radiation-data/>

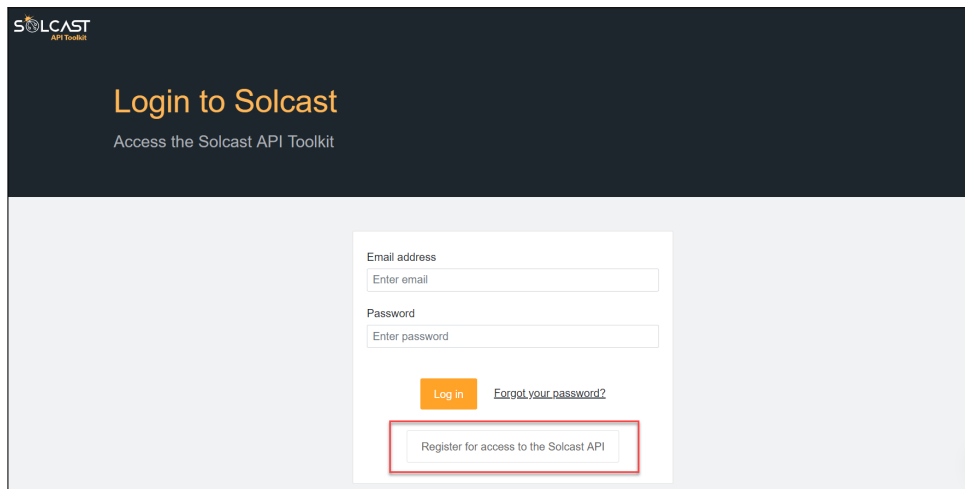
The image shows the Solcast API Toolkit login and registration page. At the top, there is a dark blue header with the Solcast logo and the text "Login to Solcast" and "Access the Solcast API Toolkit". Below the header, there is a white login form with fields for "Email address" and "Password". There are "Log in" and "Forgot your password?" links. A red box highlights the "Register for access to the Solcast API" link at the bottom of the form.

Figure 6: Solcast Login/Registration Page

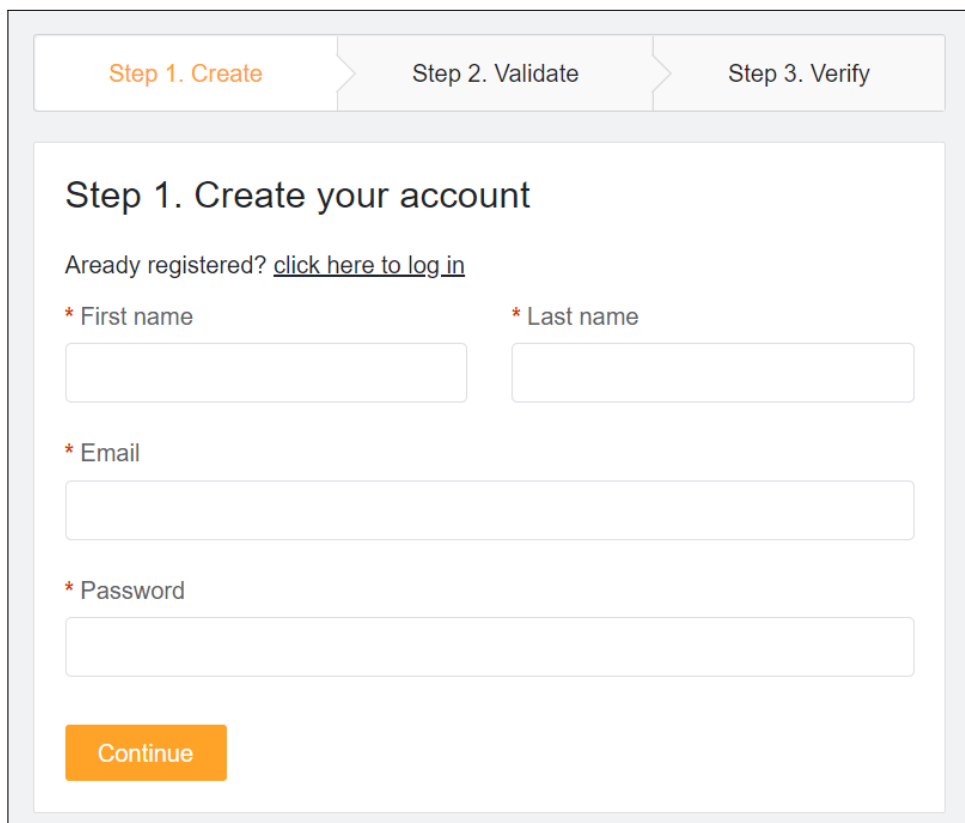
The image shows the Solcast account creation page. At the top, there are three steps: "Step 1. Create", "Step 2. Validate", and "Step 3. Verify". The main heading is "Step 1. Create your account". Below the heading, there is a link "Already registered? click here to log in". There are four required fields: "First name", "Last name", "Email", and "Password". Each field has a red asterisk indicating it is required. At the bottom, there is a "Continue" button.

Figure 7: Solcast account creation page

2. Once the account is created, login into Solcast website. You will get this screen as shown in 8 with an initial free credit of \$ 250. With each location data download for one year accounting for \$ 50. For this project the data was for one location and for three years duration so free \$150 were deducted after downloading.

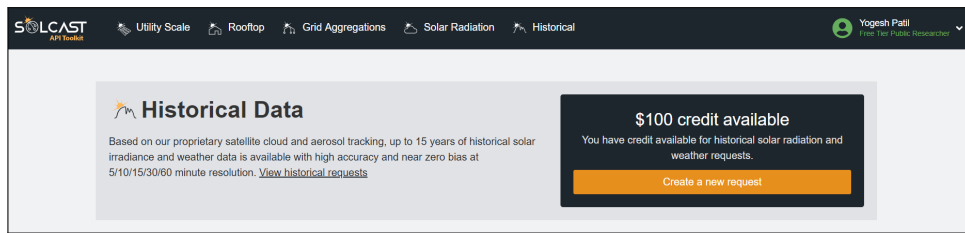


Figure 8: Solcast Free Credits for Students

3. Now to create a new historical data request follow the steps as in figures 9, 10, 11,

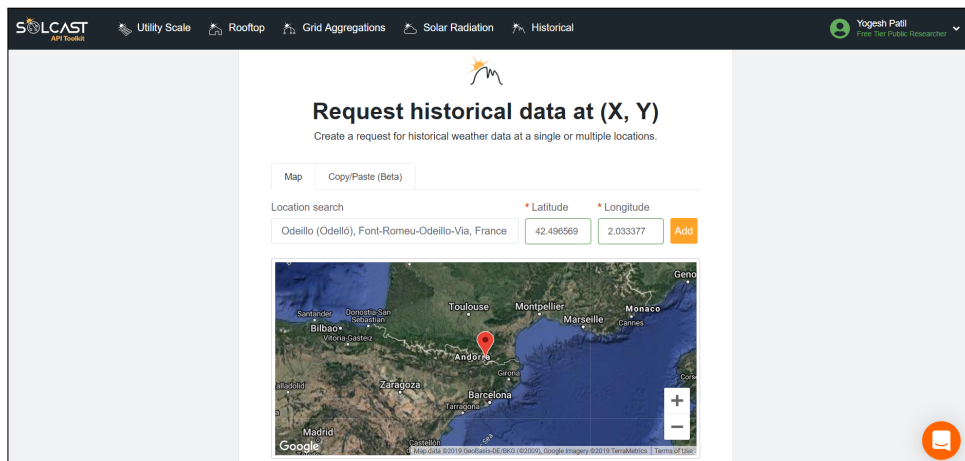


Figure 9: Historical Data Request

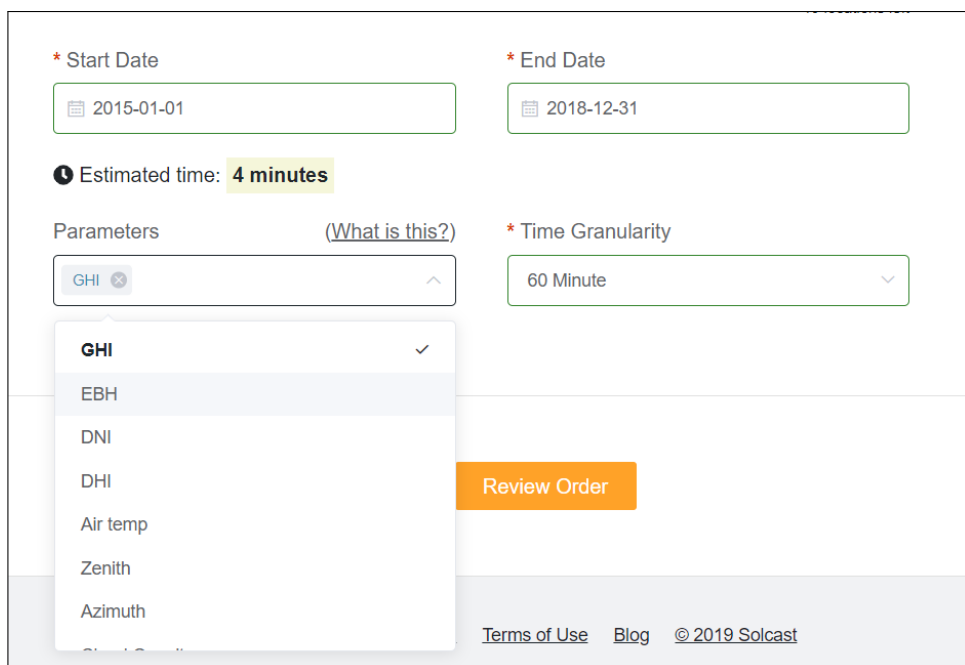


Figure 10: Solar irradiance data component selection

No.	Longitude, Latitude	Available From ?	Remove
1	2.033377, 42.496569	✓ 2008-07-20 to 2019-08-04	

19 locations left

* Start Date:

* End Date:

🕒 Estimated time: **4 minutes**

Parameters (What is this?)

GHI EBH DNI DHI Air temp Zenith Azimuth Cloud Opacity

* Time Granularity:

Figure 11: Solar irradiance data selected

4. Once the request is raised it can be checked in historical requests section as shown in figure 12. From there the data can be downloaded in csv format.

Recent Historical Requests								View all
Created	Order Id	Type	Details	Amount	Status	Actions		
2 months ago	171d9eeb-7f74-4fc7-84a2-ecca5804733d	Historical	Location: 42.4968, 2.0344 Dates: 2015-01-01 to 2018-01-02 Parameters: ghi, ebh, dni, dhi ...and 4 more	\$150.00	Ready	<button>Download</button>		

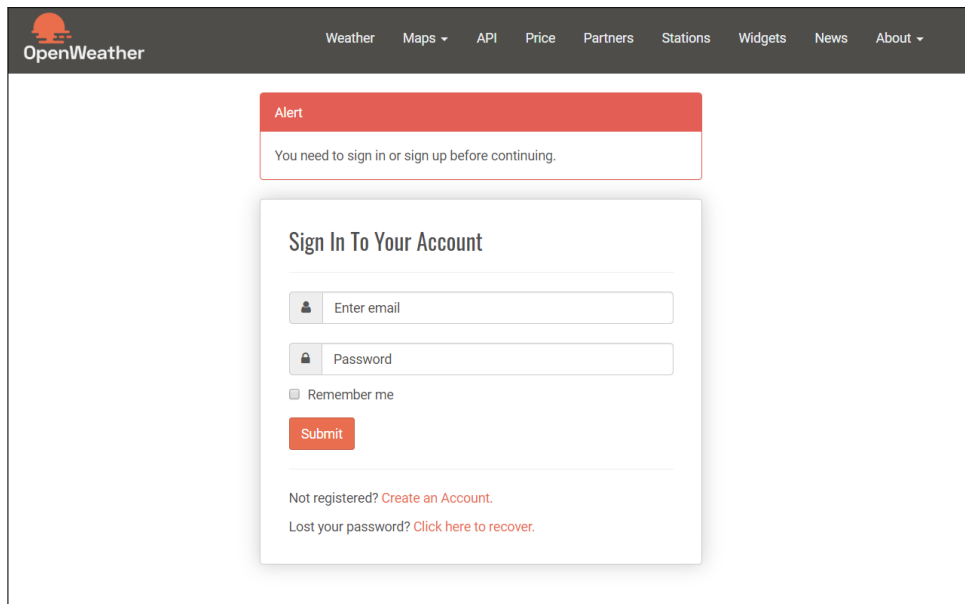
Figure 12: Recent historical Solar irradiance data downloaded

3.2 Meteorological Measures Data

For this project the meteorological measures data was downloaded from OpenWeatherMap website ⁵. The steps involved in downloading the data are as follows:

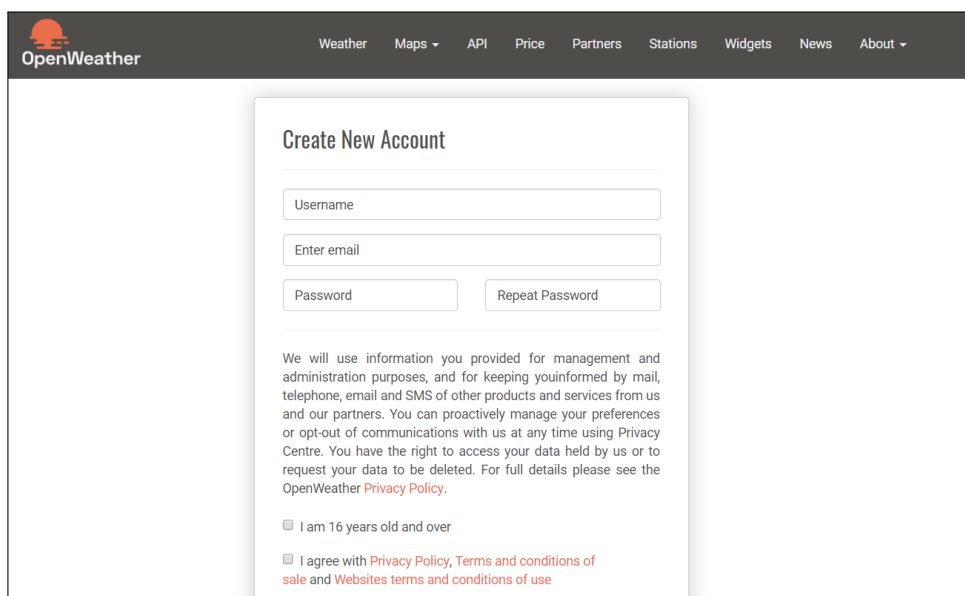
1. Go to the OpenWeatherMap website and register a new account as a student as shown in figure 6 and 7

⁵<https://openweathermap.org/>



The screenshot shows the OpenWeatherMap website's login and registration interface. At the top is a dark navigation bar with the OpenWeather logo and links for Weather, Maps, API, Price, Partners, Stations, Widgets, News, and About. Below the navigation bar, a red alert box states: "Alert: You need to sign in or sign up before continuing." The main content area features a "Sign In To Your Account" form. This form includes input fields for "Enter email" and "Password", a "Remember me" checkbox, and a red "Submit" button. Below the form, there are links for "Not registered? Create an Account." and "Lost your password? Click here to recover."

Figure 13: OpenWeatherMap Login/Registration Page



The screenshot displays the "Create New Account" page on the OpenWeatherMap website. The navigation bar is identical to the previous figure. The main form is titled "Create New Account" and contains input fields for "Username", "Enter email", "Password", and "Repeat Password". Below these fields, a paragraph of text explains the use of user information for management, administration, and communication, and provides a link to the "Privacy Policy". At the bottom of the form, there are two checkboxes: "I am 16 years old and over" and "I agree with Privacy Policy, Terms and conditions of sale and Websites terms and conditions of use".

Figure 14: OpenWeatherMap account creation page

2. Once the account is created, login into OpenWeatherMap website. You will get this screen as shown in 15. From here you can create a new historical data request do as in figure 15.

OpenWeather

Weather Maps API Price Partners Stations Widgets News About

Weather historical data available from **October 01, 2012 to August 11, 2019**. To select a date range, please use the calendar. Please notice that there is a possibility to find several locations with the same name (**duplicates**), but with the other ID. In order to get correct data please select the first one in the list.

Requests

New history bulk

* City name Search city

Available cities: **Odeillo, FR** [42.4968,2.0344] select cities

Selected cities: Odeillo, FR × [42.4968,2.0344]

* From

* To

* File format

For calculation price please fill in all fields Checkout

Figure 15: OpenWeatherMap Meteorological Data Download

4 Data Pre-processing

The data is pre-processed before putting into a model.

```

40
41 #Setting up the working directory
42 setwd("D:\\MsDataAnalytics\\Thesis\\Code")
43
44 #Read the Solar data csv
45 solar_data <- read.csv("solar_data.csv", stringsAsFactors = F)
46 str(solar_data)
47
48 #Removing T and Z from Date format in order to convert it to proper Date format
49 solar_data$PeriodEnd <- gsub("T", "", solar_data$PeriodEnd)
50 solar_data$PeriodEnd <- gsub("Z", "", solar_data$PeriodEnd)
51
52 #To separate date and time into separate columns
53 solar_data$Date <- format(as.POSIXct(solar_data$PeriodEnd,format="%Y-%m-%d %H:%M:%S"),"%Y-%m-%d")
54 solar_data$Time <- format(as.POSIXct(solar_data$PeriodEnd,format="%Y-%m-%d %H:%M:%S"),"%H")
55
56 #Convert char to Date format
57 solar_data$Date <- as.Date(solar_data$Date)
58 solar_data$Time <- as.numeric(solar_data$Time)
59 str(solar_data)
60
61 # Renaming column names
62 colnames(solar_data)[6] <- "DHI"
63 colnames(solar_data)[7] <- "DNI"
64 colnames(solar_data)[8] <- "EBH"
65 colnames(solar_data)[9] <- "GHI"
66
67 #removing the Period column
68 solar_data$Period <- NULL
69
70 #Rearranging the columns so that its more readable
71 solar_data <- solar_data[c(9,10,7,4,5,6,8,1,2,3,11)]
72

```

Figure 16: Loading and processing of Solar Data

```

Solar_ML_Code.R* x
Source on Save
Run

73 #Loading the Meteorological measures data
74 meteorological_data <- read.csv("meteorological_data.csv", stringsAsFactors = F)
75 str(meteorological_data)
76
77 #Removing the unwanted columns
78 meteorological_data$dt <- NULL
79 meteorological_data$city_id <- NULL
80 meteorological_data$city_name <- NULL
81 meteorological_data$lat <- NULL
82 meteorological_data$lon <- NULL
83 meteorological_data$temp_max <- NULL
84 meteorological_data$temp_min <- NULL
85 meteorological_data$sea_level <- NULL
86 meteorological_data$grnd_level <- NULL
87 meteorological_data$wind_deg <- NULL
88 meteorological_data$rain_1h <- NULL
89 meteorological_data$rain_3h <- NULL
90 meteorological_data$rain_24h <- NULL
91 meteorological_data$rain_today <- NULL
92 meteorological_data$snow_1h <- NULL
93 meteorological_data$snow_3h <- NULL
94 meteorological_data$snow_24h <- NULL
95 meteorological_data$snow_today <- NULL
96 meteorological_data$weather_id <- NULL
97 meteorological_data$weather_icon <- NULL
98 meteorological_data$weather_description <- NULL
99
100 #Removing the unwanted characters from dt_iso
101 meteorological_data$dt_iso <- gsub("\\+0000 UTC", "", meteorological_data$dt_iso)
102

```

Figure 17: Loading and processing of Meteorological Data

```

Solar_ML_Code.R* x
Source on Save
Run
Sc

103 #####
104 ## Merging of solar and meteorological data ##
105 #####
106
107 merged_data <- merge(solar_data, meteorological_data, by.x = "PeriodEnd", by.y = "dt_iso")
108 str(merged_data)
109 #To separate date and time into separate columns
110 merged_data$Date <- format(as.POSIXct(merged_data$PeriodEnd, format="%Y-%m-%d %H:%M:%S"), "%Y-%m-%d")
111 merged_data$Time <- format(as.POSIXct(merged_data$PeriodEnd, format="%Y-%m-%d %H:%M:%S"), "%H")
112 #Convert char to Date format
113 merged_data$Date <- as.Date(merged_data$Date)
114 merged_data$Time <- as.numeric(merged_data$Time)
115
116 merged_data$PeriodEnd <- NULL
117 merged_data$weather_main <- NULL
118 merged_data$AirTemp <- NULL
119 merged_data <- merged_data[c(2,9,14,7,13,6,3,4,5,8,10,11,12,1)]
120 #Taking data before 2018-01-01
121 merged_data <- merged_data[merged_data$Date < "2018-01-01",]
122 #Taking data after 11 am and before 6pm
123 merged_data = merged_data[merged_data$Time > 11 ,]
124 merged_data = merged_data[merged_data$Time < 18 ,]
125 #remove NA if any
126 merged_data <- na.omit(merged_data)
127 #Rename the columns properly
128 colnames(merged_data)[3] <- "Clouds"
129 colnames(merged_data)[5] <- "Wind.Speed"
130 colnames(merged_data)[8] <- "Cloud.Opacity"
131 colnames(merged_data)[11] <- "Temperature"
132 colnames(merged_data)[12] <- "Pressure"
133 colnames(merged_data)[13] <- "Humidity"
134 str(merged_data)
135 #write data to csv
136 write.csv(merged_data, 'MergedSolarData.csv', row.names=FALSE)
137

```

Figure 18: Merging of Solar and Meteorological Data

5 Data Preparation

```
Solar_ML_Code.R* x
Source on Save

366 #####
367 ## Data preparation according to time horizons ##
368 #####
369
370
371 nrow(test_data_merge_h4)
372 merged_data_h1 = merged_data[merged_data$Time < 13 ,]
373 merged_data_h2 = merged_data[merged_data$Time < 14 ,]
374 merged_data_h3 = merged_data[merged_data$Time < 15 ,]
375 merged_data_h4 = merged_data[merged_data$Time < 16 ,]
376 merged_data_h5 = merged_data[merged_data$Time < 17 ,]
377 merged_data_h6 = merged_data[merged_data$Time < 18 ,]
378
379 merged_data_h1 <- aggregate(.~Date, merged_data_h1, mean)
380 merged_data_h2 <- aggregate(.~Date, merged_data_h2, mean)
381 merged_data_h3 <- aggregate(.~Date, merged_data_h3, mean)
382 merged_data_h4 <- aggregate(.~Date, merged_data_h4, mean)
383 merged_data_h5 <- aggregate(.~Date, merged_data_h5, mean)
384 merged_data_h6 <- aggregate(.~Date, merged_data_h6, mean)
385
386 write.csv(merged_data_h1, 'merged_data_h1.csv', row.names=FALSE)
387 write.csv(merged_data_h2, 'merged_data_h2.csv', row.names=FALSE)
388 write.csv(merged_data_h3, 'merged_data_h3.csv', row.names=FALSE)
389 write.csv(merged_data_h4, 'merged_data_h4.csv', row.names=FALSE)
390 write.csv(merged_data_h5, 'merged_data_h5.csv', row.names=FALSE)
391 write.csv(merged_data_h6, 'merged_data_h6.csv', row.names=FALSE)
392
```

Figure 19: Code for data preparation for time horizon(h+1 to h+6)

```
Solar_ML_Code.R* x
Source on Save

426 #####
427 ## Data preparation for Autumn ##
428 #####
429
430 for(i in 1:6){
431   merged_data_ha<-assign(paste0("merged_data_h",i), get(paste0("merged_data_h", i)))
432   merged_data_h1_autumn9 = merged_data_ha[month(as.POSIXlt(merged_data_ha$Date, format="%d-%m-%Y")) == 9,]
433   merged_data_h1_autumn10 <- merged_data_ha[month(as.POSIXlt(merged_data_ha$Date, format="%d-%m-%Y")) == 10,]
434   merged_data_h1_autumn11 = merged_data_ha[month(as.POSIXlt(merged_data_ha$Date, format="%d-%m-%Y")) == 11,]
435
436   merged_data_h1_autumn <- rbind(merged_data_h1_autumn9,merged_data_h1_autumn10,merged_data_h1_autumn11)
437   merged_data_h1_autumn <- merged_data_h1_autumn[order(as.Date(merged_data_h1_autumn$Date, format="%d-%m-%Y")),]
438   write.csv(merged_data_h1_autumn, paste0("merged_data_autumn_h", i, ".csv"), row.names=FALSE)
439 }
440
441 #####
442 ## Data preparation for Spring ##
443 #####
444
445 for(i in 1:6){
446   merged_data_hs<-assign(paste0("merged_data_h",i), get(paste0("merged_data_h", i)))
447   merged_data_h1_spring3 = merged_data_hs[month(as.POSIXlt(merged_data_hs$Date, format="%d-%m-%Y")) == 3,]
448   merged_data_h1_spring4 <- merged_data_hs[month(as.POSIXlt(merged_data_hs$Date, format="%d-%m-%Y")) == 4,]
449   merged_data_h1_spring5 = merged_data_hs[month(as.POSIXlt(merged_data_hs$Date, format="%d-%m-%Y")) == 5,]
450
451   merged_data_h1_spring <- rbind(merged_data_h1_spring3,merged_data_h1_spring4,merged_data_h1_spring5)
452   merged_data_h1_spring <- merged_data_h1_spring[order(as.Date(merged_data_h1_spring$Date, format="%d-%m-%Y")),]
453   write.csv(merged_data_h1_spring, paste0("merged_data_spring_h", i, ".csv"), row.names=FALSE)
454 }
455
```

Figure 20: Code for data preparation for Autumn and Springs

```

394 #####
395 ## Data preparation for Winters ##
396 #####
397
398
399 for(i in 1:6){
400   merged_data_hw<-assign(paste0("merged_data_h",i), get(paste0("merged_data_h", i)))
401   merged_data_h1_winters12 = merged_data_hw[month(as.POSIXlt(merged_data_hw$Date, format="%d-%m-%Y")) == 12,]
402   merged_data_h1_winters1 <- merged_data_hw[month(as.POSIXlt(merged_data_hw$Date, format="%d-%m-%Y")) == 1,]
403   merged_data_h1_winters2 = merged_data_hw[month(as.POSIXlt(merged_data_hw$Date, format="%d-%m-%Y")) == 2,]
404
405   merged_data_h1_winters <- rbind(merged_data_h1_winters1,merged_data_h1_winters2,merged_data_h1_winters12)
406   merged_data_h1_winters <- merged_data_h1_winters[order(as.Date(merged_data_h1_winters$Date, format="%d-%m-%Y")),]
407   write.csv(merged_data_h1_winters, paste0("merged_data_winters_h", i, ".csv"), row.names=FALSE)
408 }
409
410 #####
411 ## Data preparation for Summers ##
412 #####
413
414 for(i in 1:6){
415   merged_data_hs<-assign(paste0("merged_data_h",i), get(paste0("merged_data_h", i)))
416   merged_data_h1_summers6 = merged_data_hs[month(as.POSIXlt(merged_data_hs$Date, format="%d-%m-%Y")) == 6,]
417   merged_data_h1_summers7 <- merged_data_hs[month(as.POSIXlt(merged_data_hs$Date, format="%d-%m-%Y")) == 7,]
418   merged_data_h1_summers8 = merged_data_hs[month(as.POSIXlt(merged_data_hs$Date, format="%d-%m-%Y")) == 8,]
419
420   merged_data_h1_summers <- rbind(merged_data_h1_summers6,merged_data_h1_summers7,merged_data_h1_summers8)
421   merged_data_h1_summers <- merged_data_h1_summers[order(as.Date(merged_data_h1_summers$Date, format="%d-%m-%Y")),]
422   write.csv(merged_data_h1_summers, paste0("merged_data_summers_h", i, ".csv"), row.names=FALSE)
423 }
424
425

```

Figure 21: Code for data preparation for Summers and Winters

6 Exploratory Data Analysis

```

182 #####
183 ##### Relationship between GHI and Other Predictors #####
184 #####
185 ggplot(merged_data, aes(x = Humidity, y = GHI)) +
186   geom_point(color = "steelblue") +
187   labs(x = "Humidity", y = "GHI",
188        title = "Relationship between GHI and Humidity") +
189   geom_smooth(method = "lm", se = FALSE)
190
191 ggplot(merged_data, aes(x = Zenith, y = GHI)) +
192   geom_point(color = "steelblue") +
193   labs(x = "Zenith", y = "GHI",
194        title = "Relationship between GHI and Zenith") +
195   geom_smooth(method = "lm", se = FALSE)
196
197 ggplot(merged_data, aes(x = Pressure, y = GHI)) +
198   geom_point(color = "steelblue") +
199   labs(x = "Pressure", y = "GHI",
200        title = "Relationship between GHI and Pressure") +
201   geom_smooth(method = "lm", se = FALSE)
202
203 ggplot(merged_data, aes(x = Temperature, y = GHI)) +
204   geom_point(color = "steelblue") +
205   labs(x = "Temperature", y = "GHI",
206        title = "Relationship between GHI and Temperature") +
207   geom_smooth(method = "lm", se = FALSE)
208
209 ggplot(merged_data, aes(x = Azimuth, y = GHI)) +
210   geom_point(color = "steelblue") +
211   labs(x = "Azimuth", y = "GHI",
212        title = "Relationship between GHI and Azimuth") +
213   geom_smooth(method = "lm", se = FALSE)
214
215 ggplot(merged_data, aes(x = Cloud.Opacity, y = GHI)) +
216   geom_point(color = "steelblue") +
217   labs(x = "Cloud Opacity", y = "GHI",
218        title = "Relationship between GHI and Cloud Opacity") +
219   geom_smooth(method = "lm", se = FALSE)
220

```

Figure 22: Code for Relationship between Dependent and Other Predictors (Part I)

```

220
221 ggplot(merged_data, aes(x = EBH, y = GHI)) +
222   geom_point(color = "steelblue") +
223   labs(x = "EBH", y = "GHI",
224        title = "Relationship between GHI and EBH") +
225   geom_smooth(method = "lm", se = FALSE)
226
227 ggplot(merged_data, aes(x = Wind.Speed, y = GHI)) +
228   geom_point(color = "steelblue") +
229   labs(x = "Wind Speed", y = "GHI",
230        title = "Relationship between GHI and Wind Speed") +
231   geom_smooth(method = "lm", se = FALSE)
232
233 ggplot(merged_data, aes(x = Clouds, y = GHI)) +
234   geom_point(color = "steelblue") +
235   labs(x = "Clouds", y = "GHI",
236        title = "Relationship between GHI and Clouds") +
237   geom_smooth(method = "lm", se = FALSE)
238

```

Figure 23: Code for Relationship between Dependent and Other Predictors (Part II)

```

359 #####
360 ## Correlation Matrix ##
361 #####
362
363 correlation = cor(merged_data, method = c("spearman"))
364 corplot(correlation, method = "pie")
365

```

Figure 24: Code for Correlation Matrix

```

160
161 #####
162 ### Added Variable Plots of GHI, DHI and DNI vs Predictors ###
163 #####
164
165 install.packages("car")
166 library(car)
167
168 #For GHI
169 model <- glm(GHI ~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data = merged_data, family = gaussian)
170 avPlots(model,col = "steelblue", col.lines = "#e50000",main=paste("Added Variable Plot for GHI"),marginal.scale=TRUE)
171
172 #For DHI
173 model <- glm(DHI ~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data = merged_data, family = gaussian)
174 avPlots(model,col = "steelblue", col.lines = "#e50000",main=paste("Added Variable Plot for DHI"),marginal.scale=TRUE)
175
176 #For DNI
177 model <- glm(DNI ~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data = merged_data, family = gaussian)
178 avPlots(model,col = "steelblue", col.lines = "#e50000",main=paste("Added Variable Plot for DNI"),marginal.scale=TRUE)
179

```

Figure 25: Code for Added Variable plot

7 Feature Selection

```
Solar_ML_Code.R
#####
298 # Feature Importance based on Random Forest ##
299 #####
300 #####
301 #####
302 #####
303 ###For GHI###
304 #####
305
306 merged_data_randomF = randomForest(GHI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=merged_data)
307 # Create an importance based on mean decreasing gini
308 imp <- varImpPlot(merged_data_randomF)
309 imp <- as.data.frame(imp)
310 imp$varnames <- rownames(imp)
311 rownames(imp) <- NULL
312
313 ggplot(imp, aes(x=reorder(varnames, IncNodePurity), y=IncNodePurity, color=as.factor(varnames))) +
314   geom_point(size=3) +
315   geom_segment(aes(x=varnames, xend=varnames, y=0, yend=IncNodePurity)) +
316   ylab("IncNodePurity") + theme(axis.text = black.bold.8.text) +
317   xlab("Variable Name") + theme(axis.text = black.bold.8.text) +
318   coord_flip()
319
320 #####
321 ###For DHI###
322 #####
323 merged_data_randomF = randomForest(DHI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=merged_data)
324 # Create an importance based on mean decreasing gini
325 imp <- varImpPlot(merged_data_randomF)
326
327 imp <- as.data.frame(imp)
328 imp$varnames <- rownames(imp)
329 rownames(imp) <- NULL
330
331 ggplot(imp, aes(x=reorder(varnames, IncNodePurity), y=IncNodePurity, color=as.factor(varnames))) +
332   geom_point(size=3) +
333   geom_segment(aes(x=varnames, xend=varnames, y=0, yend=IncNodePurity)) +
334   ylab("IncNodePurity") + theme(axis.text = black.bold.8.text) +
335   xlab("Variable Name") + theme(axis.text = black.bold.8.text) +
336   coord_flip()
337
338
```

Figure 26: Code for feature selection using random forest (Part I)

```
Solar_ML_Code.R
339 #####
340 ###For DNI###
341 #####
342
343 merged_data_randomF = randomForest(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=merged_data)
344 # Create an importance based on mean decreasing gini
345 imp <- varImpPlot(merged_data_randomF)
346
347 imp <- as.data.frame(imp)
348 imp$varnames <- rownames(imp)
349 rownames(imp) <- NULL
350
351 ggplot(imp, aes(x=reorder(varnames, IncNodePurity), y=IncNodePurity, color=as.factor(varnames))) +
352   geom_point(size=3) +
353   geom_segment(aes(x=varnames, xend=varnames, y=0, yend=IncNodePurity)) +
354   ylab("IncNodePurity") + theme(axis.text = black.bold.8.text) +
355   xlab("Variable Name") + theme(axis.text = black.bold.8.text) +
356   coord_flip()
357
358
```

Figure 27: Code for feature selection using random forest (Part II)

```

239
240 - #####
241 - ##### Feature selection using Boruta Algorithm #####
242 - #####
243 summary(merged_data)
244 set.seed(111)
245
246 #For GHI
247 merged_data_train_GHI <- Boruta(GHI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data = merged_data, doTrace = 2)
248 print(merged_data_train_GHI)
249 merged_boruta_GHI <- names(merged_data_train_GHI$finalDecision[merged_data_train_GHI$finalDecision %in% c("Confirmed", "Tentative")])
250 print(merged_boruta_GHI)
251 plot(merged_data_train_GHI, cex.axis=.7, las=2, xlab="", main="Variable Importance for GHI")
252 merged_data_train_GHI_df <- attStats(merged_data_train_GHI)
253 print(merged_data_train_GHI_df)
254 merged_data_train_GHI_df <- merged_data_train_GHI_df[c(-3,-4,-5)]
255 merged_data_train_GHI_df$minImp <- NULL
256 merged_data_train_GHI_df$maxImp <- NULL
257 merged_data_train_GHI_df$normHits <- NULL
258 summary(merged_data_train_GHI_df)
259 write.csv(merged_data_train_GHI_df, 'merged_data_train_GHI_df.csv')
260
261 #For DHI
262 merged_data_train_DHI <- Boruta(DHI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data = merged_data, doTrace = 2)
263 print(merged_data_train_DHI)
264 merged_boruta_DHI <- names(merged_data_train_DHI$finalDecision[merged_data_train_DHI$finalDecision %in% c("Confirmed", "Tentative")])
265 print(merged_boruta_DHI)
266 plot(merged_data_train_DHI, cex.axis=.7, las=2, xlab="", main="Variable Importance for DHI")
267 merged_data_train_DHI_df <- attStats(merged_data_train_DHI)
268 merged_data_train_DHI_df <- merged_data_train_DHI_df[c(-3,-4,-5)]
269 write.csv(merged_data_train_DHI_df, 'merged_data_train_DHI_df.csv')
270
271 #For DNI
272 merged_data_train_DNI <- Boruta(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data = merged_data, doTrace = 2)
273 print(merged_data_train_DNI)
274 merged_boruta_DNI <- names(merged_data_train_DNI$finalDecision[merged_data_train_DNI$finalDecision %in% c("Confirmed", "Tentative")])
275 print(merged_boruta_DNI)
276 ggplot(merged_data_train_DNI, cex.axis=.7, las=2, xlab="", main="Variable Importance for DNI")
277 merged_data_train_DNI_df <- attStats(merged_data_train_DNI)
278 merged_data_train_DNI_df <- merged_data_train_DNI_df[c(-3,-4,-5)]
279 write.csv(merged_data_train_DNI_df, 'merged_data_train_DNI_df.csv')
280

```

Figure 28: Code for feature selection using Boruta Algorithm

```

281 - #####
282 - ### VIF- For multicollinearity check###
283 - #####
284
285 xx_reg <- lm(GHI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=merged_data)
286 summary(xx_reg)
287 vif(xx_reg)
288 #if greater than 5 then it should be discarded
289
290 xx_reg <- lm(DHI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=merged_data)
291 summary(xx_reg)
292 vif(xx_reg)
293
294 xx_reg <- lm(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=merged_data)
295 summary(xx_reg)
296 vif(xx_reg)
297

```

Figure 29: Code for feature selection using VIF

8 Implementation, Evaluation and Results

```

455
456 #####
457 ##### Test and Train data Creation #####
458 #####
459
460 merged_data_trial <- merged_data
461 summary(merged_data_trial)
462 merged_data_trial$GHI
463
464 merged_data_trial$GHI[merged_data_trial$GHI == 0] <- NA
465 merged_data_trial$GHI <- na_mean(merged_data_trial$GHI)
466 set.seed(1)
467
468 #Data split
469 div <- sample.split(merged_data_trial, SplitRatio = 0.8)
470 TrainData <- subset(merged_data_trial, div == TRUE)
471 TestData <- subset(merged_data_trial, div == FALSE)
472
473 dim(TestData)
474
475

```

Figure 30: Code for test and train data creation

```

478 ## Performance Metrics Calculation Loop ###
479 #Creation of Data Frame with Table for Evaluation Metrics for all models and all horizons
480 perf_metric_dataframe <- data.frame(matrix(ncol = 8, nrow = 24), stringsAsFactors=FALSE)
481 x <- c("Metric", "Model", "h+1", "h+2", "h+3", "h+4", "h+5", "h+6")
482 colnames(perf_metric_dataframe) <- x
483 #Creation of Function to put the results in the Data frame by time horizons
484 setresults <- function(model, rmse, mae, nrmse, nmae){
485   if(sum(is.na(perf_metric_dataframe$h+1')) != 0){
486     i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h+1'))-1))
487     perf_metric_dataframe$Metric[i] <- "RMSE"
488     perf_metric_dataframe$Model[i] <- model
489     perf_metric_dataframe$Metric[i+1] <- "MAE"
490     perf_metric_dataframe$Model[i+1] <- model
491     perf_metric_dataframe$Metric[i+2] <- "nRMSE"
492     perf_metric_dataframe$Model[i+2] <- model
493     perf_metric_dataframe$Metric[i+3] <- "nMAE"
494     perf_metric_dataframe$Model[i+3] <- model
495     perf_metric_dataframe$h+1[i] <- rmse
496     perf_metric_dataframe$h+1[i+1] <- mae
497     perf_metric_dataframe$h+1[i+2] <- nrmse
498     perf_metric_dataframe$h+1[i+3] <- nmae
499   } else if(sum(is.na(perf_metric_dataframe$h+2')) != 0){
500     i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h+2'))-1))
501     perf_metric_dataframe$h+2[i] <- rmse
502     perf_metric_dataframe$h+2[i+1] <- mae
503     perf_metric_dataframe$h+2[i+2] <- nrmse
504     perf_metric_dataframe$h+2[i+3] <- nmae
505   } else if(sum(is.na(perf_metric_dataframe$h+3')) != 0){
506     i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h+3'))-1))
507     perf_metric_dataframe$h+3[i] <- rmse
508     perf_metric_dataframe$h+3[i+1] <- mae
509     perf_metric_dataframe$h+3[i+2] <- nrmse
510     perf_metric_dataframe$h+3[i+3] <- nmae
511   } else if(sum(is.na(perf_metric_dataframe$h+4')) != 0){
512     i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h+4'))-1))
513     perf_metric_dataframe$h+4[i] <- rmse
514     perf_metric_dataframe$h+4[i+1] <- mae
515     perf_metric_dataframe$h+4[i+2] <- nrmse
516     perf_metric_dataframe$h+4[i+3] <- nmae
517   } else if(sum(is.na(perf_metric_dataframe$h+5')) != 0){
518     i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h+5'))-1))
519     perf_metric_dataframe$h+5[i] <- rmse

```

Figure 31: Code for loading performance metrics in tabular form (Part I)

```

516 perf_metric_dataframe$h4[i+3] <- nmae
517 } else if(sum(is.na(perf_metric_dataframe$h5`)) !=0){
518 i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h5`))-1))
519 perf_metric_dataframe$h5[i] <- rmse
520 perf_metric_dataframe$h5[i+1] <- mae
521 perf_metric_dataframe$h5[i+2] <- nrmse
522 perf_metric_dataframe$h5[i+3] <- nmae
523 } else if(sum(is.na(perf_metric_dataframe$h6`)) !=0){
524 i <- as.numeric(nrow(perf_metric_dataframe) - (sum(is.na(perf_metric_dataframe$h6`))-1))
525 perf_metric_dataframe$h6[i] <- rmse
526 perf_metric_dataframe$h6[i+1] <- mae
527 perf_metric_dataframe$h6[i+2] <- nrmse
528 perf_metric_dataframe$h6[i+3] <- nmae
529 }
530 print(perf_metric_dataframe)
531 }
532 }
533

```

Figure 32: Code for loading performance metrics in tabular form (Part II)

```

541
542 for (j in 1:6){
543
544 ## For ANNUAL ##
545 merged_data_main<-assign(paste0("merged_data_h",j), get(paste0("merged_data_h", j)))
546 # div <- sample.split(merged_data_main, SplitRatio = 0.8)
547 # TrainData <- subset(merged_data_main, div == TRUE)
548 # TestData <- subset(merged_data_main, div == FALSE)
549 # dim(TestData)
550
551 ## For Seasons ##
552 seasons_main <- read.csv(paste0("merged_data_h",j,".csv"), stringsAsFactors = F)
553 div <- sample.split(seasons_main, SplitRatio = 0.8)
554 TrainData <- subset(seasons_main, div == TRUE)
555 TestData <- subset(seasons_main, div == FALSE)
556
557 # Run algorithms using 10-fold cross validation
558 control <- trainControl(method="cv", number=10)
559
560 #####
561 ##### CART #####
562 #####
563
564 set.seed(7)
565 CART_Train <- train(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=TrainData, method="rpart", trControl=control)
566 predicted_val_cart <- predict(CART_Train ,TestData)
567
568 #Evaluation Metrics
569 cart_rmse<- hydroGOF::rmse(predicted_val_cart,TestData$DNI)
570 cart_nrmse<- hydroGOF::nrmse(predicted_val_cart,TestData$DNI,norm = "sd")
571 cart_mae<- hydroGOF::mae(predicted_val_cart,TestData$DNI)
572 cart_nmae <- compute.nmae(predicted_val_cart,TestData$DNI)*100
573
574 #Function call to append the results
575 perf_metric_dataframe <- setresults("CART", cart_rmse, cart_mae, cart_nrmse, cart_nmae)
576

```

Figure 33: Code for Loading of Data and CART model implementation

```

578 #####
579 ##### LINEAR REGRESSION #####
580 #####
581 summary(TrainData)
582
583 LR_Train <- train(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=TrainData, method="lm", trControl=control)
584 predicted_val_LR <- predict(LR_Train ,TestData)
585
586 ## Evaluation Metrics
587
588 LR_rmse<- hydroGOF::rmse(predicted_val_LR,TestData$DNI)
589 LR_nrmse<- hydroGOF::nrmse(predicted_val_LR,TestData$DNI,norm = "sd")
590 LR_mae<- hydroGOF::mae(predicted_val_LR,TestData$DNI)
591 LR_nmae <- compute.nmae(predicted_val_LR,TestData$DNI)*100
592
593 #Function call to append the results
594 perf_metric_dataframe <- setresults("LR", LR_rmse, LR_mae, LR_nrmse, LR_nmae)
595
596 #####
597 ##### Stochastic Gradient Boosting #####
598 #####
599
600 SGB_Train <- train(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=TrainData, method="gbm", trControl=control)
601 predicted_val_SGB <- predict(SGB_Train ,TestData)
602
603 ## Evaluation Metrics
604 SGB_rmse<- hydroGOF::rmse(predicted_val_SGB,TestData$DNI)
605 SGB_nrmse<- hydroGOF::nrmse(predicted_val_SGB,TestData$DNI,norm = "sd")
606 SGB_mae<- hydroGOF::mae(predicted_val_SGB,TestData$DNI)
607 SGB_nmae <- compute.nmae(predicted_val_SGB,TestData$DNI)*100
608
609 #Function call to append the results
610 perf_metric_dataframe <- setresults("SGB", SGB_rmse, SGB_mae, SGB_nrmse, SGB_nmae)
611
612

```

Figure 34: Code for implementation of LR and SGB model

```

613- ##### KNN #####
614- #####
615- #####
616-
617- tuneGrid<- expand.grid(k=9)
618-
619- set.seed(7)
620- KNN_Train <- train(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=TrainData, method="knn",trControl=control)
621- predicted_val_knn <- predict(KNN_Train ,TestData)
622-
623- ## Evaluation Metrics
624- KNN_rmse<- hydroGOF::rmse(predicted_val_knn,TestData$DNI)
625- KNN_nrmse<- hydroGOF::nrmse(predicted_val_knn,TestData$DNI,norm = "sd")
626- KNN_mae<- hydroGOF::mae(predicted_val_knn,TestData$DNI)
627- KNN_nmae <- compute.nmae(predicted_val_knn,TestData$DNI)*100
628-
629- #Function call to append the results
630- perf_metric_dataframe <- setresults("KNN", KNN_rmse, KNN_mae, KNN_nrmse, KNN_nmae)
631-
632- ##### SVM #####
633- #####
634- #####
635- #####
636-
637- set.seed(7)
638- SVM_Train <- train(DNI~ Humidity+Zenith+Pressure+Temperature+Cloud.Opacity+Azimuth+Wind.Speed+Clouds, data=TrainData, method="svmRadial",trControl=control)
639-
640- predicted_val_svm <- predict(SVM_Train ,TestData)
641-
642- #Evaluation Metrics
643- SVM_rmse<- hydroGOF::rmse(predicted_val_svm,TestData$DNI)
644- SVM_nrmse<- hydroGOF::nrmse(predicted_val_svm,TestData$DNI,norm = "sd")
645- SVM_mae<- hydroGOF::mae(predicted_val_svm,TestData$DNI)
646- SVM_nmae <- compute.nmae(predicted_val_svm,TestData$DNI)*100
647-
648- #Function call to append the results
649- perf_metric_dataframe <- setresults("SVM", SVM_rmse, SVM_mae, SVM_nrmse, SVM_nmae)
650-
651-

```

Figure 35: Code for implementation of KNN and SVM model

```

652- ##### RANDOM FOREST #####
653- #####
654- #####
655- #####
656-
657- set.seed(7)
658- tuneGrid <- expand.grid(.mtry=7)
659-
660- RF_Train <- train(GHI~GHI, data=TrainData, method="rf", tuneGrid= tuneGrid,trControl=control,prox=TRUE,allowParallel=TRUE)
661- predicted_val_rf <- predict(RF_Train ,TestData)
662-
663- #Evaluation Metrics
664- RF_rmse<- hydroGOF::rmse(predicted_val_rf,TestData$DNI)
665- RF_nrmse<- hydroGOF::nrmse(predicted_val_rf,TestData$DNI,norm = "sd")
666- RF_mae<- hydroGOF::mae(predicted_val_rf,TestData$DNI)
667- RF_nmae <- compute.nmae(predicted_val_rf,TestData$DNI)*100
668-
669- #Function call to append the results
670- perf_metric_dataframe <- setresults("RF", RF_rmse, RF_mae, RF_nrmse, RF_nmae)
671-
672-
673- }
674-
675-

```

Figure 36: Code for implementation of RF model

```

677-
678- #For ANNUAL
679- Eval_metric_GHI_Annual <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
680- write.csv(Eval_metric_GHI_Annual, 'Eval_metric_GHI_Annual.csv', row.names=FALSE)
681-
682- Eval_metric_DHI_Annual <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
683- write.csv(Eval_metric_DHI_Annual, 'Eval_metric_DHI_Annual.csv', row.names=FALSE)
684-
685- Eval_metric_DNI_Annual <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
686- write.csv(Eval_metric_DNI_Annual, 'Eval_metric_DNI_Annual.csv', row.names=FALSE)
687-
688-

```

Figure 37: Code for saving annual evaluation metrics file

```

689 #For SEASONS
690 #WINTERS
691 Eval_metric_GHI_Winters <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
692 write.csv(Eval_metric_GHI_Winters, 'Eval_metric_GHI_Winters.csv', row.names=FALSE)
693
694 Eval_metric_DHI_Winters <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
695 write.csv(Eval_metric_DHI_Winters, 'Eval_metric_DHI_Winters.csv', row.names=FALSE)
696
697 Eval_metric_DNI_Winters <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
698 write.csv(Eval_metric_DNI_Winters, 'Eval_metric_DNI_Winters.csv', row.names=FALSE)
699
700 #For SUMMERS
701 Eval_metric_GHI_Summers <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
702 write.csv(Eval_metric_GHI_Summers, 'Eval_metric_GHI_Summers.csv', row.names=FALSE)
703
704 Eval_metric_DHI_Summers <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
705 write.csv(Eval_metric_DHI_Summers, 'Eval_metric_DHI_Summers.csv', row.names=FALSE)
706
707 Eval_metric_DNI_Summers <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
708 write.csv(Eval_metric_DNI_Summers, 'Eval_metric_DNI_Summers.csv', row.names=FALSE)
709
710 #For Autums
711 Eval_metric_GHI_Autumn <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
712 write.csv(Eval_metric_GHI_Autumn, 'Eval_metric_GHI_Autumn.csv', row.names=FALSE)
713
714 Eval_metric_DHI_Autumn <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
715 write.csv(Eval_metric_DHI_Autumn, 'Eval_metric_DHI_Autumn.csv', row.names=FALSE)
716
717 Eval_metric_DNI_Autumn <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
718 write.csv(Eval_metric_DNI_Autumn, 'Eval_metric_DNI_Autumn.csv', row.names=FALSE)
719
720 #For Springs
721 Eval_metric_GHI_Spring <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
722 write.csv(Eval_metric_GHI_Spring, 'Eval_metric_GHI_Spring.csv', row.names=FALSE)
723
724 Eval_metric_DHI_Spring <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
725 write.csv(Eval_metric_DHI_Spring, 'Eval_metric_DHI_Spring.csv', row.names=FALSE)
726
727 Eval_metric_DNI_Spring <- perf_metric_dataframe[order(perf_metric_dataframe$Metric),]
728 write.csv(Eval_metric_DNI_Spring, 'Eval_metric_DNI_Spring.csv', row.names=FALSE)
729

```

Figure 38: Code for saving seasonal evaluation metrics file

9 Visualization of Results in Tableau

The files with results generated in 38 are visualized in Tableau.

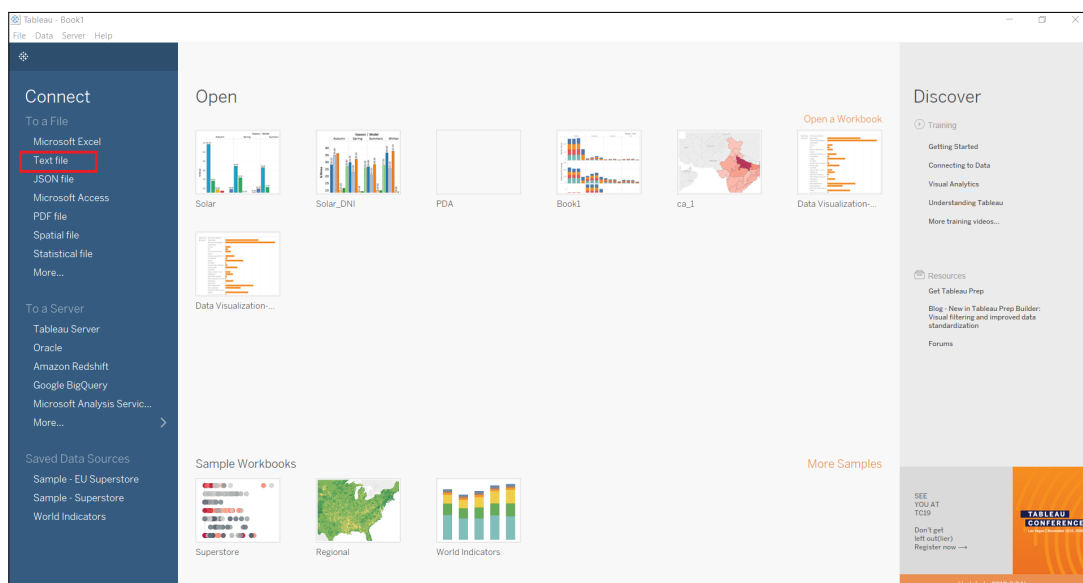


Figure 39: Uploading results csv into Tableau

Tableau - Book1

File Data Server Window Help

Connections: Seasonal_GHI (Text file)

Files: Use Data Interpreter (checked), Data Interpreter might be able to clean your Text file workbook.

Seasonal_GHI.csv

Sort fields: Data source order

Show aliases Show hidden fields 48 rows

Model	Horizon	Component	Season	N_Rmse
CART	h+1	GHI	Autumn	48.5000
LR	h+1	GHI	Autumn	16.6000
SGB	h+1	GHI	Autumn	14.2000
KNIN	h+1	GHI	Autumn	28.1000
SVM	h+1	GHI	Autumn	24.7000
RF	h+1	GHI	Autumn	11.8000
CART	h+6	GHI	Autumn	68.3000
LR	h+6	GHI	Autumn	18.8000
SGB	h+6	GHI	Autumn	12.3000
KNIN	h+6	GHI	Autumn	28.1000
SVM	h+6	GHI	Autumn	19.0000

Figure 40: Screenshot after loading the result file in Tableau

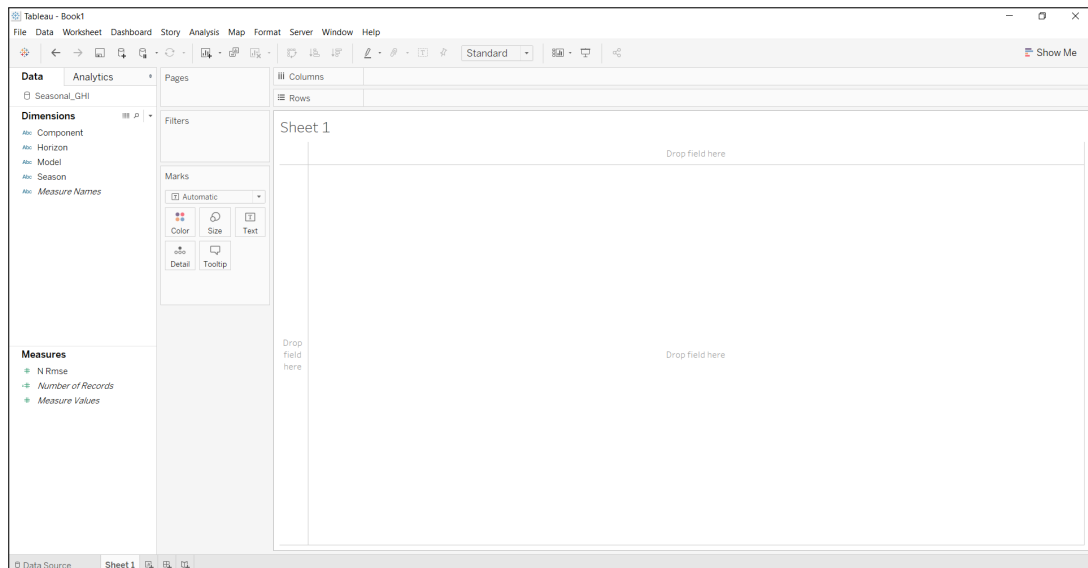


Figure 41: Screenshot of Sheet1 after loading data

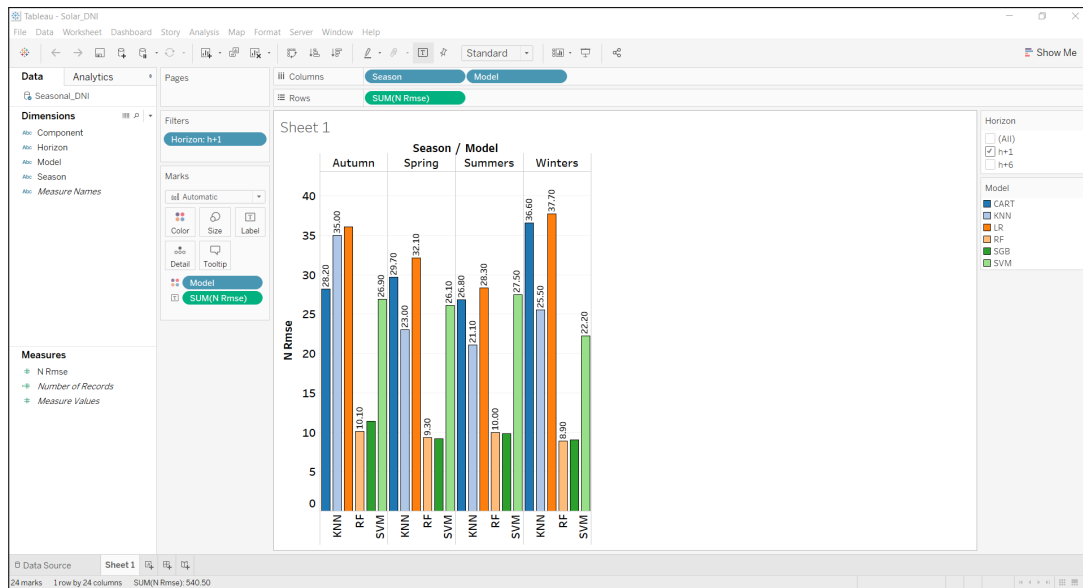


Figure 42: Screenshot after arranging the data into rows and columns