Comparative analysis of Machine learning Algorithms for Estimating Global Solar Radiation at Selected Weather Stations in Vhembe District Municipality.

By

Marandela Mulalo Valencia

11629239

Supervisor(s):

Dr. TS Mulaudzi

Dr. NE Maluta

A mini dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science Degree in the field of e-Science

Faculty of Science, Engineering and Agriculture

Department of Mathematics and Computational Sciences

University of Venda

Thohoyandou,Limpopo

South Africa

Declaration

I, **Marandela Mulalo Valencia of student number 11629239**, declare that this research titled: Comparative analysis of Machine learning Algorithms for Estimating Global Solar Radiation at Selected Weather Stations in Vhembe District Municipality, submitted in partial fulfilment of the requirements for the conferral of the degree Master of e-Science, from the University of Venda, is my work and that all the sources that I have used or quoted have been indicated and acknowledged using complete references.

I further declare that I have not previously submitted this work, or part of it, for examination at University of Venda for another qualification or at any other higher education institution.

..... 06 June 2023

Ms. M.V Marandela (Student) Date

Acknowledgements

Firstly, I would like to extend my sincere gratitude to Dr E Maluta and Dr Mulaudzi NS for being my supervisors and taking me through this research, my family and friends who gave their moral support and lastly the DST-CSIR National e-Science Postgraduate Teaching and Training Platform which provided funding.

Sponsors: The National e-Science Postgraduate Teaching and Training Platform(NEPTTP).

Abstract

Estimating and assessing the energy falling in a particular area is essential for installers of renewable technologies. Different equations have been applied as the most reliable empirical for estimating global solar radiation(GSR) in different climatic conditions. The main objective of this work is to estimate the global solar radiation of two stations namely, Mutale and Messina found in Vhembe District, Limpopo Province, South Africa. Four different methods (Random forest(RF) regression, K-nearest neighour (K-NN), Support Vector Machines(SVM) and Extreme Gradient Boosting mechanism(XGBoost)) is used to estimate the GRS in this study. The RF model on Mutale station was found to be the best fitting model with $R^2 = 0.9902$, $MSE = 0.4085$ and $RMSE = 0.6391$, followed by XGB with $R^2 = 0.9898$, $MSE = 0.4245$ and $RMSE = 0.6515$. RF was also found to be the best for Messina station with $R^2 = 0.9636$, $MSE = 0.1.4138$ and $RMSE = 1.1890$, followed by XGB model with $R^2 = 0.9595$, $MSE = 1.5723$ and $RMSE = 1.2539$. From the results, it can be concluded that RF is a better model for estimating GSR for different stations.

Keywords: Machine Learning, Empirical models, Random Forest, Support Vector Mechanism, Artificial Neural Networks, Decision Tree, Linear regression.

Contents

List of Figures

List of Tables

List of Abbreviations

GSR	**G**lobal **S**olar **R**adiation
ML	**M**achine **L**earning
RMSE	**R**oot **M**ean **S**quare **E**rror
ANN	**A**rtificial **N**eural **N**etwork
MLR	**M**ulti-Layer **P**erception
RBF	**R**adial **B**ass **F**unctions
BPANN	**B**ack **P**ropagation **A**rtificial **N**eural **N**etwork
SR	**S**olar **R**adiation
R^2	**R-squared**
SVM	**S**upport **V**ector **M**achine
H_0	**E**xtraterrestrial **R**adiation
I_{sc}	**s**olar **C**onstant
σ	**D**eclination **A**ngle
D_n	**J**ulian **C**alender **D**ay
ω_s	**H**our **A**ngle
S_0	**D**ay **l**ength

Chapter 1

Introduction

1.1 Introduction

The role of energy in industrial revolutions can never be questioned. Energy is in the centre of every revolution, those who were ahead of the curve have reaped significant economic benefits [4]. South African energy supplier(Eskom) is experiencing a challenge in supplying energy demand, resulting in a contraction of the economy. South Africa is in a state of emergency with its energy supply hence the implementation of load shedding. Energy is at the centre of economic activity and where there is inefficient energy supply ,limited economic activity, underdevelopment grabs society, life expectancy takes a dive, all because of energy. Developing countries' scramble to get economic activity going through other forms of energy, i.e., coal, oil and nuclear, has had a devastating effect on the environment.

The environment is being ravaged by climate change. As a result, energy producers, scientists, and governments have been pushed to reconsider their energy production strategies. Renewable energy has been the focus to reduce the effects of climate change and also developing countries are looking at it because of the costs associated with it and for the stability it provides in energy generation [7]. This study will investigate how to harness Global Solar radiation efficiently and methods to remove clutter in the data collected. This done because the effects of climate change cannot not be ignored any longer.

1.2 Background

The estimate of global solar radiation using various machine learning methods has been the subject of several research initiatives. This section is a review of a sample of work on the topic. Previous relevant work, as well as the linked approaches used, is outlined in the section below, as well as a discussion of issues that have been solved using both machine learning and conventional methods techniques.

Understanding the type and distribution of incoming radiation, and the climate conditions observed in that region, requires knowledge of the available solar resources at a particular site [6].

Estimation of solar radiation models that are accurate and efficient are continually being pursued, created and improved due to the limited number of meteorological stations that actively record solar radiation data [6]. It is important to note that inaccurate models can cause significant variations in the estimation of GSR and as a result , the assessment of a PV system's yearly energy production will be inconsistent [6].

1.3 Problem Statement

With global warming being an issue and the use of non-renewable resources slowly depleting , countries are looking into maximizing renewable energy and also being economical about it. Developing countries such as South Africa do not have enough prediction models or they cannot afford to have them in every station.

The established artificial intelligence models estimate the solar radiation time series data more effectively compared to the traditional procedures based on clearness index [2]. Since not every area or solar station are able to measure their global solar radiation, the prediction will help those areas where measurements are not available.

1.4 Research Questions

With the use of machine learning, which model performs best in estimating the global solar radiation? Can the prediction help others stations to predict GSR so that they can optimize effective renewable energy distribution?

1.5 Research Aim and Objectives

1.5.1 Research Aim

- To access the potential of ML models for modelling GSR from parameters collected in the two stations.
- To develop ML simulations that predict the amount of solar radiation received using two solar stations.

1.5.2 Objectives

The objectives of the research are to:

- Determining the best fit parameters for the model using data that has been collected daily.
- Develop different machine learning models to predict the global solar radiation.
- Train models from the features given from the supplied dataset and compare, analyse and evaluate different models to determine the one that performs best using mean square error(MSE),root mean square error(RMSE) and r-squared(R^2).

1.5.3 Benefits

Solar power systems can optimize the distribution of energy efficiently. Prediction assists in in 'developing and administering' photo-voltaic systems while also delivering various economic benefits to energy suppliers. The modeling will generate global solar irradiation predictions at different sites where the measured data is unavailable.

1.5.4 Importance of the study

Generation of electricity with coal in South Africa contribute to global warming. To mitigate this challenge, the country needs to protect the environment. So, renewable energy plays a crucial role. It is of importance to know the amount of solar radiation that can be harnessed at a particular area. Due to lack of weather stations that measure these data, prediction of the such data can become handy.

South Africa's solar potential is among the greatest in the world, with an average of 2500 hours of sunlight each year, including plenty of sunshine throughout the winter months [12, 5]. Each province receives different amounts of solar radiation, with an average radiation intensity of 4:5 to 6:6 kWh=m2 [12, 5]. In this paper, we study the estimation of global radiation using data that is collected in the Vhembe district area in Limpopo Province. With sunshine accessible throughout the year, Vhembe is ideally suited for solar energy harvesting. Our research attempts to estimate the amount of untapped solar potential in the Vhembe district while also attempting to improve solar technology awareness and implementation.

Different machine learning models will be developed to predict the global solar radiation. This study is important because it grasps the gap of integration of solar energy and the ability to properly manage the transition between intermittent and conventional energy.

Unlike traditional production of energy technologies, which are unaffected by weather circumstances, solar energy integration is extremely dependent on weather conditions [4].

Solar energy integration into electrical networks needs reliable forecast information of solar resources, allowing it to efficiently quantify available energy and manage the transition between intermittent and conventional points [10].

1.6 Overview

The rest of the study is divided into four chapters. Chapter 2, discusses relevant topics in the field of study. Chapter 3 gives the information of how the data will be retrieved, cleaned, and analysed, Chapter 3 also discusses the methods methods in modeling data and measuring their errors. Chapter 4 gives the results of all the models and models comparison. Chapter 5 provides a conclusion with, recommends, and areas for future studies.

Chapter 2

Literature Review

2.1 Empirical models

Marzo et al. [11] estimated daily global solar radiation in desert areas using Artificial Neural Networks (ANN), where the inputs used were daily minimum and maximum temperatures and extraterrestrial radiation. On the study, the ANN model was validated with data from deserts in Chile, Israel, Saudi Arabia, South Africa and Australia. The results showed that the average Relative Root-Mean-Square Deviation (RRMSD) value was 13%, the average Relative Mean Bias Difference (RMBE) value was less than 4% and the average correlation coefficient (r) value was about 0.8.

Mulaudzi, Sankaran, and Lysko [13], studied the estimate the global solar radiation from the observed mean daily maximum and minimum air temperatures for some selected stations in the Vhembe District and tested the validity of the theoretical equation by Angstrom [1]. Further more, the Angstrom-Prescott linear regression model for Vhembe District was represented by:

$$\frac{\bar{H}}{H_0} = 0.2 + 0.5.\frac{\bar{N}_a}{N_p} \tag{2.1}$$

where

- 0.2 and 0.5 are the regression coefficients,
- $\bar{H}_0$ is the monthly average horizontal extraterrestrial solar radiation
- $\bar{N}_a$ and N_p are daily average duration of the actual and predicted possible sunshine hours respectively .

The estimated horizontal global solar radiation of each station in each station in the Vhembe region was then compared with the measured values. It was concluded that the monthly average daily global solar radiation on a horizontal surface for the Vhembe Region may be

estimated using the re-parameterized Angström-Prescott linear regression given above [13].

2.2 Machine learning applications

Machine learning is a data analysis technique that facilitates the creation of analytical models. It is a sub-field of artificial intelligence predicated on the premise that algorithms can learn from data, spot trends, and make different choices with little or no human interaction.

Artificial Neural Network Library is a free, open source neural network library, that implements multi layer artificial neural networks with support for fully and sparsely connected networks .

Back-propagation is a way of propagating the total loss back into the neural network to know how much loss every node is responsible for, and subsequently updating the weights to minimize the loss by giving the nodes with higher error rates lower weights and vice versa. The advantage of BPANN comes from its amazing information processing characteristics applicable mainly to non linearity, adaptability, increased parallelism, learning capability, and fault and noise tolerance [14].

Machine learning approaches have been successfully applied in the field of new energy for predictive modeling and capture of long and short-term features of non-linear time series [17]. The paper by Guermoui et al. [7], evaluated the performance of artificial neural network(ANN) which proves to be one of the earliest machine learning method that is commonly used to estimate and predict solar radiation but it has a problem of instability with short data and has a tendency of over fitting the training data.

Guermoui et al. [7] and Maluta, Mulaudzi, and Sankaran [10], used two methods which were Multi-Layer perception and Radial Basis functions(RBF) on estimating the global solar radiation in eight different stations and deduced that in terms of RMSE between the observed and predicted solar radiation values. The two methods performed very well but their results concluded that the Multi-Layer perception model performed better than the physical model. In this paper, It was also discovered that MLP models that utilize air temperature parameters outperform those that use relative humidity, implying that combining air temperature and relative humidity does not significantly increase model performance.

The most significant finding was that employing sunlight ratio alone as an input parameter results in excellent performance due to its great connection with clearness index. Combining solar duration with air temperature and relative humidity improves performance and model accuracy when mean temperatures, rainfall, and humidity are used as input parameters [7].

Brkić and Tuka [2] used an empirical model for estimating solar radiation based on air temperature in three different areas in Europe where monthly average daily GSR data was essential in the design. In the first area [2] proposed theoretical model for estimating global solar radiation based on sunshine duration [9] and [15] revised the model to allow calculation of monthly average daily global radiation (MJ/m2 day) on a horizontal surface from monthly average daily total insolation on an extraterrestrial horizontal surface using the following relationship. The monthly average daily global radiation on horizontal surface:

$$\frac{\bar{H}}{\bar{H_0}} = (a+b)\frac{\bar{S}}{\bar{S_0}} \tag{2.2}$$

where, $\bar{H}$ is the monthly average daily extraterrestrial radiation on horizontal surface, $\bar{S}$ is the monthly average of daily sunshine duration. Where the equations of $\bar{S_0}$ and $\bar{H_0}$ are given by:

$$\begin{aligned} \bar{S_0} &= \tfrac{2}{15} arccos(-tan\varphi tan\delta) &= \frac{2}{\omega_s} \\ \bar{H_0} &= \tfrac{24.3600.G_{sc}}{\pi}[1+0.003cos(\tfrac{360}{365}n)].(cos\varphi cos\delta cos\omega_s + \tfrac{\pi\omega_s}{180} sin\varphi sin\delta). \end{aligned}$$

Younas investigated the SR in Ethopia using the ANN method. Climatological and meteorological parameters were important parameters indicated the amount of SR in Addis Ababa town [2]. Backward propagation artificial Neural network (BPANN) and ANN models were used to input independent variables (daily sunshine, humidity, wind speed,solar radiation and maximum and minimum temperature) [2].

Although the meteorological data had some missed values, data imputation technique was used to replace the missing data. Performance evaluation of the development of BPANN and ANN models was used using the Root Mean Square Error(RMSE). RMSE indicates data on temporary performance and measure the difference of predicted value in the area of the actual measured data[5]. If the RMSE is low, it means that the estimation SR is more accurate. From the results, it was found that BPANN model achieves best performance with a correlation coefficient of R=0.901.

In the early 1980's, Hargreaves developed an equation to estimate SR for calculating the relations:

$$\begin{aligned} H_0 &= \frac{24}{\pi} * I_{sc} * (w_s in\phi sin\sigma + cos\pi cos\sigma sin_s) * d_r \\ \frac{\bar{H}}{\bar{H}_0} &= a.(T_{max} - T_{min})^{0.5} \end{aligned}$$

Where

$$\begin{aligned} d_r &= (1 + 0.033cos\frac{360*d}{365}) \\ w_s &= cos^{-1}(-tan\phi tan\sigma) \\ \sigma &= 23.4sin(\frac{360*(284+d)}{365}) \\ N &= \frac{2}{15} * w_s \end{aligned}$$

- T_{max} and T_{min} is the maximum and minimum daily air temperature,
- I_{sc} is solar constant,
- w_s is hour angle,
- dr is the inverse relative distance of the sun to earth,
- H_0 is monthly mean daily extraterrestrial solar radiation,
- σ is the solar declination,
- and ϕ is the latitude of the site under consideration.

Here , the GSR was also studied based on air temperatures and compared the model with other existing models at Sarajevo.From the models,it was deduced that the daily maximum temperature decreased with the increase of cloud cover.

Performance of the predictions of each model against the measured values of the monthly means of daily SR were assessed using the RMSE. The RMSE equation is:

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}, \tag{2.3}$$

where y_i is the estimated value and x_i is the measured value.

The coefficient of determination given by:

$$R^2 = \frac{[\sum_{i=1}^{X}(D_i - \bar{D}_i)(F_i - \bar{F}_i)]^2}{\sqrt{\sum_{i=1}^{X}(D_i - \bar{D}_i)^2 . \sum_{i=1}^{X}(F_i - \bar{F}_i)^2}} \tag{2.4}$$

Nash-sutcliffe efficiency index given by:

$$NSE = 1 - \frac{\sum_{i=1}^{X}(D_i - F_I)^2}{\sum_{i=1}^{X}(\bar{D}_i - D_i)^2} \tag{2.5}$$

Similar studies have been conducted for Iran where they used two support vector regressions (which are radial basis function and polynomial basis function) models and empirical models. Maximum hourly sunshine was used as their input parameter. Both methods were explored and statistical comparison was used to check which method had better predictions. It was deduced that the SVR performed better than the empirical method. Two models within the SVR and SVR radial basis function outperformed best. The statistical results were:mean absolute error percentage 0.4466%, mean absolute bias error, $1.2524\frac{MJ}{m^2}$, root mean square error 2.0046 $\frac{MJ}{m^2}$, relative root mean square error 9.0343% and coefficient determination 0.9133, respectively. Also, onmonthly mean estimation the values were 1.4078%, 0.2845 $\frac{MJ}{m^2}$, $0.45044\frac{MJ}{m^2}$, 2.2576% which showed that it was the best method to predict the solar radiation [5].

Chegaar, Lamri, and Chibani [3] used different models to study the monthly average global solar radiation. Artificial network models was the one that performed better than other models that were employed. The study was based on sunshine hour data. The relationship between sunshine duration and the amount of global solar radiation was calculated using:

$$\frac{G}{G_0} = a' + (1 - a')\frac{n}{N} \tag{2.6}$$

where, G is the mean daily total global solar radiation,G_0 is the daily global radiation, N is the maximum possible sunshine duration , n is mean daily sunshine duration, a' is mean proportion of radiation received on a completely overcast day [12].

Maluta, Mulaudzi, and Sankaran [10] studied the estimate the global solar radiation from the observed mean daily maximum and minimum air temperatures for some selected stations in the Vhembe District and to test the validity of the theoretical equation by Hargreaves and Samani.The computed values were compared with the weather station data. The empirical coefficient, Kr is also calculate and compared with the the one in the literature. The yearly average value of Kr for each station was calculated and a value of 0.16 was obtained, which was equal to the value reported on the literature. A reasonable agreement

between the estimated and observed solar radiation values suggests that this equation can be used to estimate a set of radiation data for this region.

Mulaudzi, Sankaran, and Lysko [13], given the scarcity of observable and dependable data for sun irradiance in rural South Africa, an Angström-Prescott linear correlation equation was employed to estimate the regression coefficients in the Vhembe District.

Nguyen et al. [14] estimated the SR in Turkey. The input parameters used were the astronomical factor, extraterrestrial radiation and climatic variables, sunshine duration, cloud cover, minimum temperature and maximum temperature and SR was obtained as a result.

2.3 Overview

In this research, chapter two focuses on some related work has been conducted before by other researchers were they compared different machine learning algorithms and check which one performs better. In the next chapter, we look at a number of research methods applied on the study. The methods will be accessed and compared to see which one performs best in estimating GSR.

Chapter 3

Research Methodology

This chapter provides a summary of the research techniques used in the study and training of the prediction model. It will also go through some of the most often used models in data science.

3.1 Data and studied area

3.1.1 Data and studied area

The area which is of interest in our study is Messina: NoordGrens station and Thohoyandou: Mutale station which are both in the Vhembe District ,Limpopo , South Africa. Mutale and Messina areas are known to be one of the hot areas . It has long summers and very short winters, and temperatures reach the 40 degrees, which means there it receives more than sufficient solar radiation. The figure below displays the Vhembe district from the South African map.

FIGURE 3.1: Vhembe district ,South Africa -From Wikipedia, the free encyclopedia

The data was collected for a period from 2007-2020 with average daily minimum and maximum temperatures , average relative humidity , sunshine hours , average daily wind speed and total daily rainfall.

3.1.2 Prepossessing

Variable selection by Correlation

correlation is it slightly easier to understand and explain quite simply on a scale of -1 to 1. With 0 being completely uncorrelated , -1 being the perfect negative correlation and 1 perfect positive correlation .
Here is a correlation guide:

- 0- No linear relationship
- +0.30- A weak uphill (positive) linear relationship
- +0.50- A moderate uphill (positive) relationship
- +0.70- A strong uphill (positive) linear relationship
- +1- A perfect uphill (positive) linear relationship

and vice versa.

The figure below shows the correlations in our data. It is confident to say that temperature has a positive impact on radiation while humidity affects it negatively . We can conclude that temperature plays the most vital role in maximising the solar radiation.

FIGURE 3.2: Correlation

Variable selection by significance

Sparse model will be used to determine which variables are worth keeping and which variables are worth discarding from our dataset. So the variables we end up keeping through whatever model we choose to use are the most important in helping to predict the response variable value ; that is they have the highest predictive power . Therefore the first message to take a look at takes each predict variable individually and uses it to predict the response variable. In each case the correlation between the two is measured, so between the predictor variable and the response variable or between X_1 and Y for instance and then between X_2 and y and so on ,and then the statistical significance of the predictor variable is calculated . We set a null hypothesis which has no effect on the response and we said about determine what the probability is of observing the values of the predictor variable that we have observed assuming that the null hypothesis is true. so the lower the probability is of observing the values that we have actually observed the most significant that variable is above which we are happy to discard them.

3.1.3 Train and split

To train these models, we want to use as much of our dataset as feasible. The rationale for this is that, in most cases, the more observations we have, the better.The solution to this problem of not wanting to guard into the real world and collect more data but also wanting to use as much of our existing dataset as much as possible for the training process is to use some small portion of our data to simulate the real world.

To simulate that unseen data on which the model will ultimately perform its predictions. We split the dataset into two parts. One part of the useful training the model and the other remaining part will be used for testing the model. During training model will be shown each observation in the training dateset as well as the response variable value so it will be showing all the axes(x) and it will be showing the y values for each row. During testing however the test dataset will be used so we have the response variables values for each observation we have the y values but the train model will be shown only the predictor variables for these observations but only showing the X_1 and not Y. The predictions will then be done using X-values and then be compared to the known y values, if the true values are closer to the prediction,the model will be a good model.

3.2 Methods

The utilization of renewable energy technologies relies heavily on accurate estimation and assessment of the energy potential in a given area. In the case of solar energy, determining the global solar radiation (GSR) is crucial for installers to gauge the feasibility and performance of solar installations. Over the years, researchers have developed various empirical equations as reliable tools for estimating GSR under different climatic conditions.

This study focuses on estimating the GSR for two specific stations, Mutale and Messina, located in the Vhembe District of the Limpopo Province in South Africa. The primary objective is to compare and evaluate the performance of four different estimation methods: Random Forest (RF) regression, K-nearest neighbor (K-NN), Support Vector Machines (SVM), and Extreme Gradient Boosting (XGBoost) mechanism.

3.2.1 XGBoost

Figure 3.3 shows XGBoost process.

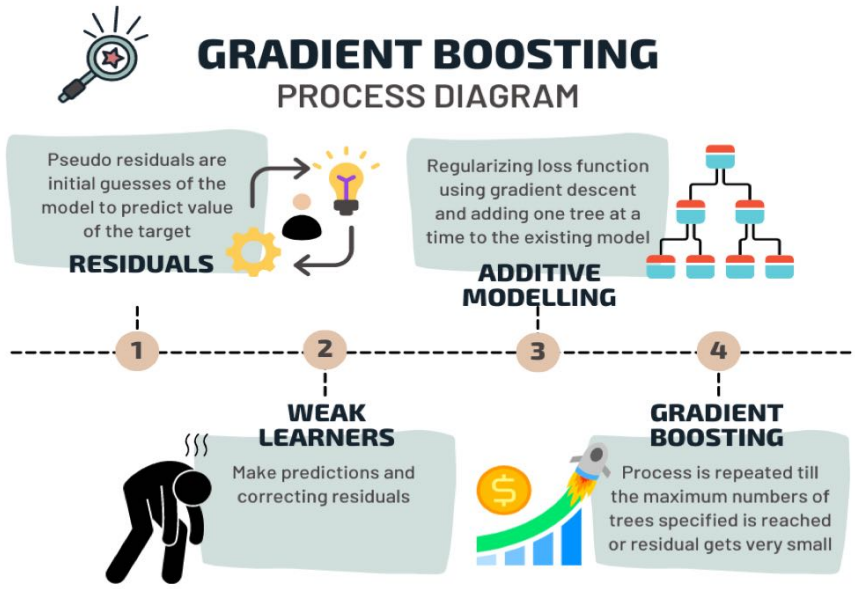


FIGURE 3.3: XGBoost process.

The extreme gradient boosting method (XGBoost) is a scalable machine learning system for tree boosting that is an improved distributed gradient boosting library that can rapidly

assess the value of all input attributes. It has shown to be a trustworthy and efficient machine learning problem solution [8]. When compared to other gradient boosting methods, XGBoost can generate a robust classifier from a collection of weak classifiers and has the following advantages: (1) handle missing values correctly; (2) prevent overfitting; and (3) parallel and distributed calculations improve processing time. The goal of XGBoost is to design and optimize the objective function by using a gradient descent optimization approach and arbitrary differentiable loss functions to minimize the loss function by adding weak learners.

This technique generates decision trees in a sequential fashion. Weights are very significant in XGBoost. All of the independent variables are given weights, which are subsequently put into the decision tree, which predicts results. The weight of factors that the tree predicted incorrectly is raised, and these variables are subsequently put into the second decision tree. These various classifiers/predictors are then combined to form a more powerful and precise model. It can solve issues including regression, classification, ranking, and user-defined prediction.

XGBoost attempts to minimize the regularized objective as follows:

$$obj(\theta) = \sum_{i=1}^{n} L(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k), f_k EF_K \tag{3.1}$$

where, K is the number of trees, f is the functional space of F, F is the set of possible CARTs. And the first term is the loss function and the second is the regularization parameter.

3.2.2 Random Forest

Random forest regression is a supervised learning algorithm that uses ensemble learning method for regression [3].The Ensemble Learning Approach is a technique that blends several machine learning algorithms with projections to make a forecast more precise than a single model [3]. Below is a simple diagram that shows a simple random forest regression structure where different trees run in parallel without contact between each other.One has a freedom of choosing how many they want to build. In jupyter notebook,libraries were imported and the dataset was loaded,the following steps where followed in order to achieve our goal:

- Y variable(dependent variable) was identified as the Radiation and X variable(Independent variable) was time

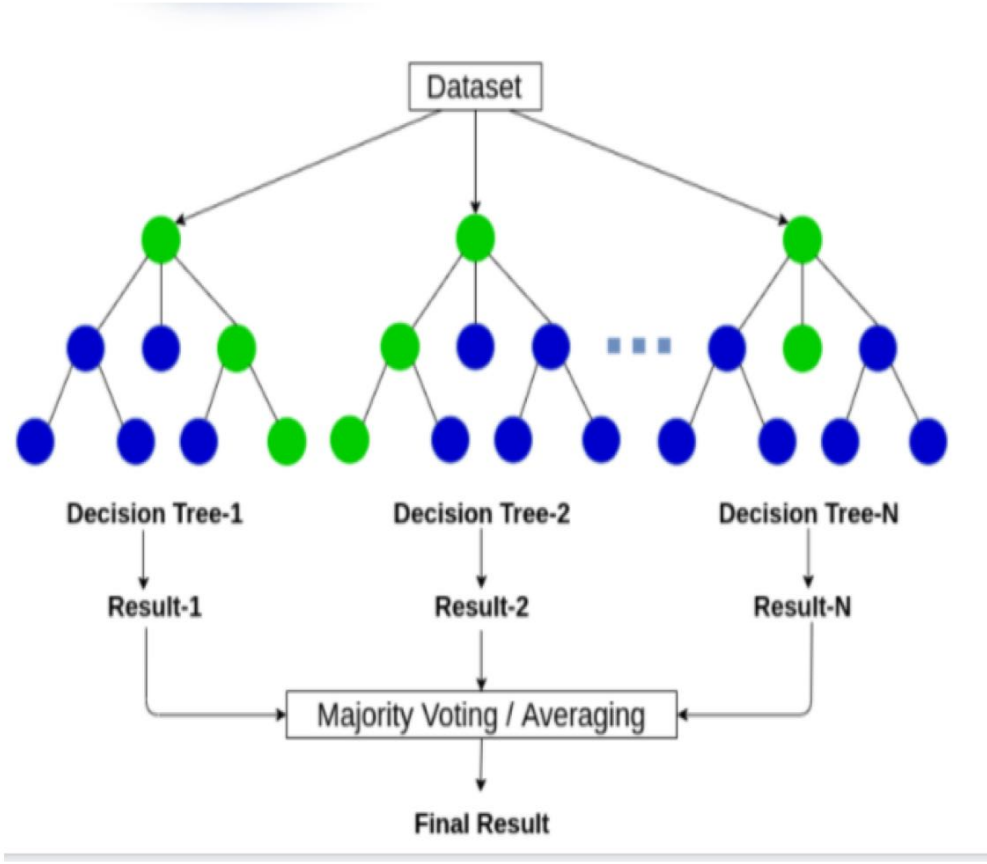


FIGURE 3.4: Random forest regression

- The data was split into training and testing sets (split the data to 85% of training and 15% of testing)
- Use random forest regression model to train the entire data.
- Use random forest regression model to test.
- Access the performance of the model using R^2.

3.2.3 K-Nearest Neighbor

The K nearest neighbors method is a simple algorithm that saves all available data and predicts the numerical target using a similarity metric (e.g., distance functions). KNN has been utilized as a non-parametric approach in statistical estimates and pattern recognition since the early 1970s.

The average of the numerical targets of the K nearest neighbors is a basic implementation of KNN regression. Another method is to take an inverse distance weighted average of the K closest neighbors. The same distance functions are used in KNN regression as in KNN classification.

3.2.4 Support Vector Mechanism

Support Vector Machine is a supervised machine learning algorithm capable of performing classification, regression and even outlier detection. The good thing about the SVM is that it solves both linear and non-linear problems.The technique generates a hyperplane that divides the data into classes. It is well known for its accuracy and Works well on smaller cleaner datasets and it can be more efficient because it uses a subset of training points.

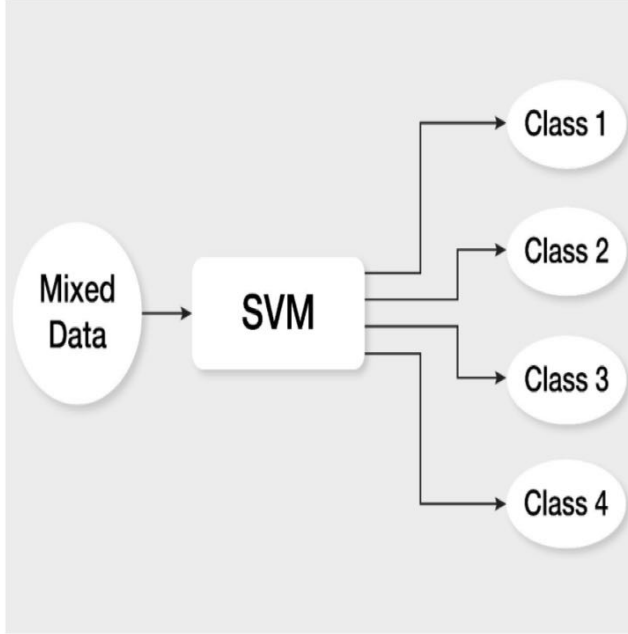


FIGURE 3.5: Support Vector Mechanism

Figure 3.5 illustrates how SVM operates when been fed with data.

3.3 Statistical measures of comparison.

The prediction results will be evaluated using the measures of errors the root mean square error (RMSE), R-squared (R^2), and the mean square error (MSE). In order to find the best model which will be used for predictions. The smaller the RMSE and MSE, the closer the

predicted return the better the fit of the model, and the closer R^2 to 1 the better the model.

- R-squared (R^2) - It ensures that no prejudices(biased) that may be embedded in the data are taken into account.It reveals how often these variables with relation to the total variables the model has predicted. It ranges from zero to positive one. If the (R^2) is close to one, we can deduce that there is strongly good linear relationship between the prediction and actual values and if it is close to zero,it means that there is less sometimes no linear relationship [13].
- Metrics square error (MSE) -It has been used because there was a lot of outliers in our data.
- Root mean square error (RMSE)-It determines the precision of the models by calculating the difference between the predicted and actual data and should always be positive.

3.4 Expected Outcome

- The aim of this task is to predict the GSR for the two stations.
- The estimation of GSR depends highly on weather conditions of the studied area i.e geographical weather parameters play a crucial role in determining efficient SR so results may differ from station to station.

Chapter 4

Results and Discussions

4.1 Introduction

The chapter presents and analyses data obtained from https://sauran.ac.za//. The results on a 1 hour signal will be provided. The analysis takes into consideration the objectives outlined in chapter 1 and methodology discussed in Chapter 3. Python 3.7 was used for analysing the GSR signal.

4.2 Data collection and description

The study uses primary 1 hour GSR signal data obtained from https://sauran.ac.za// 2006 to 2020 for Mutale station and from 2004 to 2020 for Messina station.

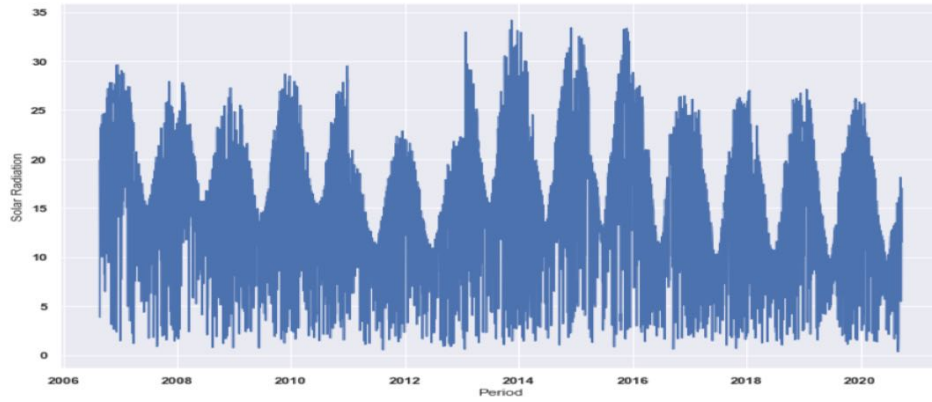
4.3 Explanatory data analysis

Explanatory data analysis (EDA) is an approach of analysing data sets and summarising their main characteristics or a critical step in analysing the data from an experiment [16]. Exploratory data analysis covers preliminary data selection of appropriate models, examinations of assumptions, and assistance to statisticians in data exploration.

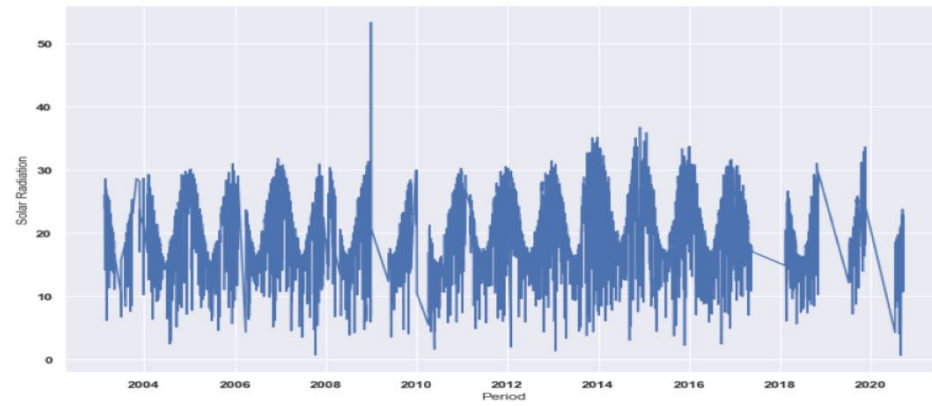
EDA is a fundamental and crucial step in the analysis of data sets. It involves exploring and understanding the main characteristics of the data through various statistical and visual techniques. EDA helps to uncover patterns, relationships, anomalies, and insights within the data, allowing researchers to gain a comprehensive understanding of the data before applying more complex models or statistical techniques.

4.3.1 Time series plots

Time series charts study data patterns and behavior across time and are frequently used to investigate the daily, weekly, and seasonal consequences of a process modification. They provide a visual representation of the time series. Time series graphs make it simple to examine data patterns.



(A) Solar Radiation for Mutale station over time.



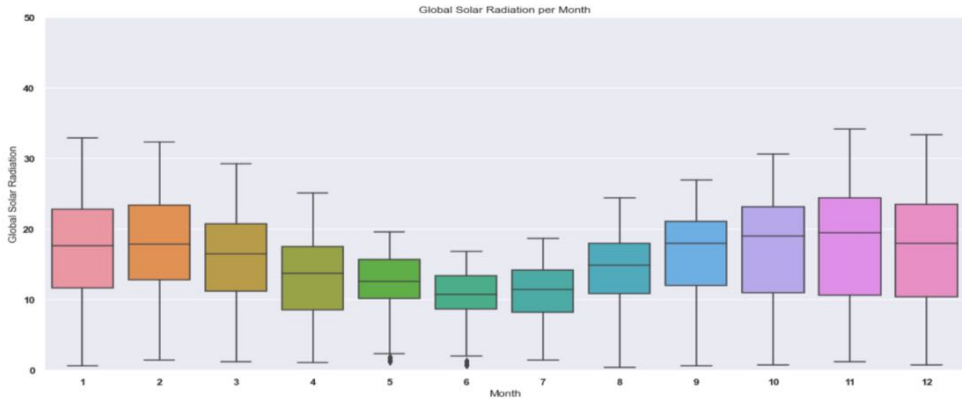
(B) Solar Radiation for Messina station over time.

FIGURE 4.1: Time series plots of GSR vs time for 2 sites.

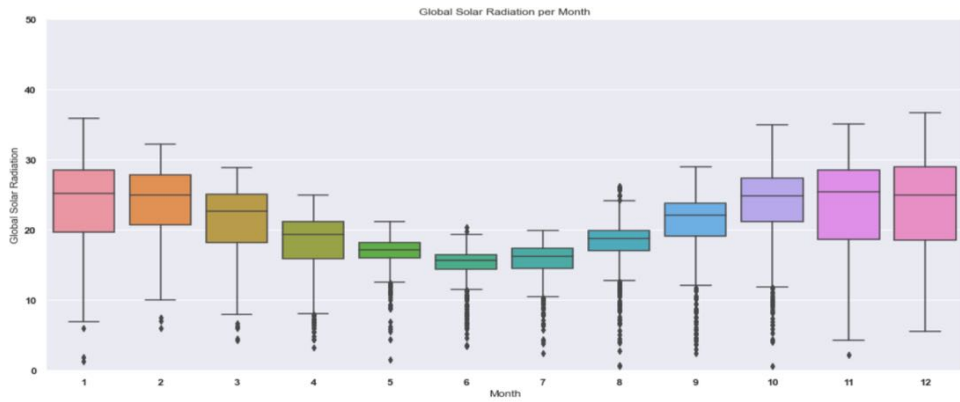
Figure 4.1 (A) gives the GSR time series plot for Mutale station, and Figure 4.1 (B) gives the GSR time series plot for the Messina station. All the graphs in Figure 4.1 show high volatilities, more GSR highs in summer, lows in winter and some consolidation in Autum and spring. During a day, the highest is recorded from all stations. The gaps in Figure 4.1 (B) shows that there were days where in the data was not recorded for the day.

4.3.2 Box plots

Box plots are particularly useful for comparing distributions or visualizing the spread and skewness of data across different categories or groups. They allow for quick visual comparisons of medians, ranges, and the presence of outliers. Additionally, box plots can reveal any asymmetry or departure from a normal distribution.



(A) Solar Radiation for Mutale station over time.



(B) Solar Radiation for Messina station over time.

FIGURE 4.2: Box plots of GSR vs time for 2 sites.

Figure 4.2 (A) gives the GSR box plot for Mutale station, and Figure 4.2 (B) gives the GSR time series plot for the Messina station. As to confirm the results from Figure 4.1, and Figure 4.2 shows high volatilities (box plots are more spread), more GSR highs in summer (months 12-3), lows in winter(month 6-9) and some consolidation in Autum(month 3-6) and spring(month 9-12).

4.3.3 Summary of descriptive statistics

Table 4.1 reports the key descriptive statistics of the GSR series. Descriptive statistics quantitatively describe or summarize features of the collected information. Descriptive statistics provides sample summaries about the sample. The sample summaries are the minimum, mean, maximum, standard deviations, skewness, kurtosis and sample size.

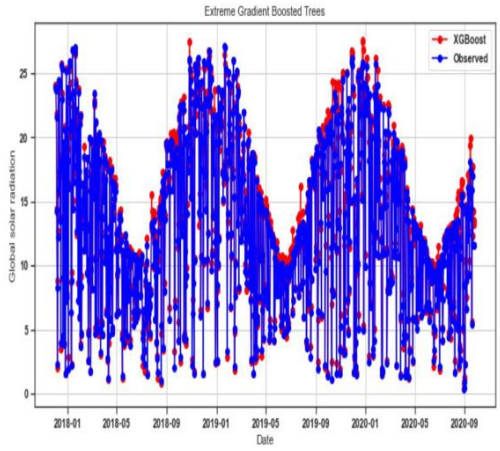
TABLE 4.1: 1 hour GSR data for 2 stations descriptive statistics.

Variables	Mutale	Messina
Min	0.34	0.58
Mean	14.82	19.93
Std Dev	6.95	5.97
Kurtosis	-0.629	-0.043
Skewness	0.042	-0.056
Max	34.18	56.36
n	5136	4836

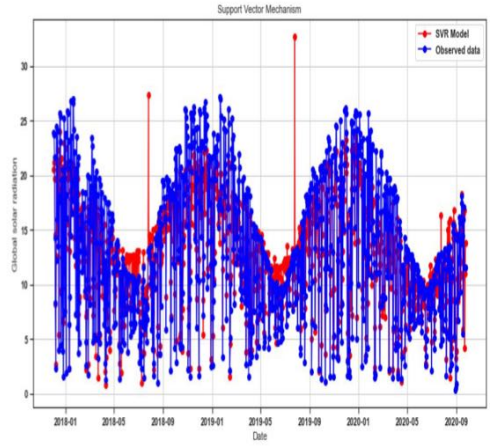
The kurtosis is a statistical measure used to describe the distribution and can also help describe the distribution's tails in relation to the overall shape. Kurtosis has three categories which can be displayed by a data set, i.e. mesokurtic(kurt=3), leptokurtic(kurt>3), and platykurtic(kurt<3). The GSR signals have kurtosis values less than 3 for all three sites. Thus, they are platykurtic, implying that both the distributions produce fewer and less severe outliers.

4.4 Models comparisons

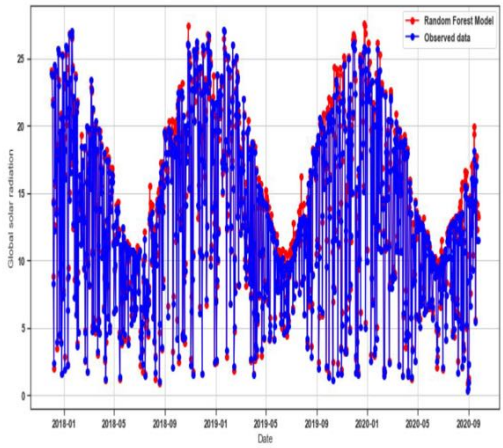
4.4.1 Mutale Station



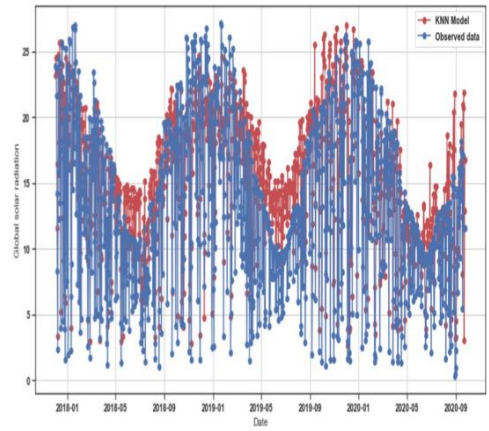
(A) XGBoost



(B) Support Vector Mechanism



(C) Random Forest

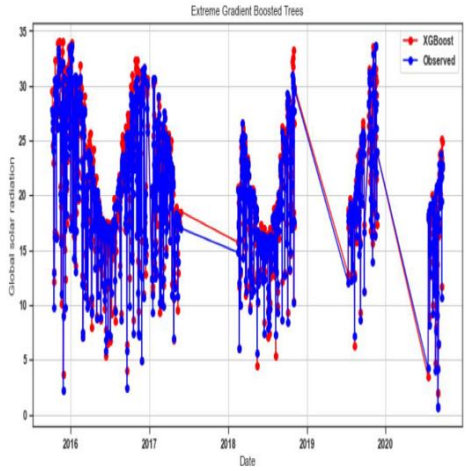


(D) K-NN

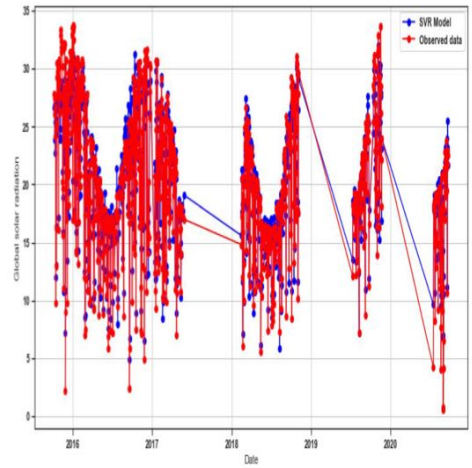
FIGURE 4.3: Graphical representation of model comparisons for Mutale Station.

From Figures 4.3 (A)-(D), the red lines represents the observed vales and the blue lines represents the predicted values for a specific model. It can also be seen that in all the figures that the observed values are much closer to the predicted values on Random Forest model, followed by the XGBoost, SVM and then K-NN.

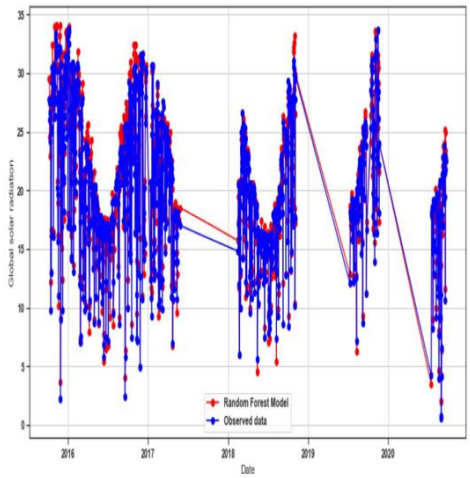
4.4.2 Musina Station



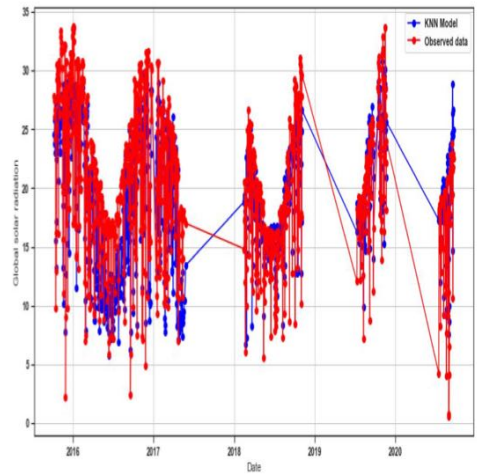
(A) XGBoost



(B) SVM



(C) Random Forest



(D) K-NN

FIGURE 4.4: Support Vector Mechanism

From Figures 4.4 (A)-(D), the red lines represents the observed vales and the blue lines represents the predicted values for a specific model. It can also be seen that in all the figures that the observed values are much closer to the predicted values on Random Forest model, followed by the XGBoost, SVM and then K-NN.

4.4.3 Comparison of model performance using statistical measures

TABLE 4.2: Performance comparison of station for each algorithm.

Station	Metric	XGB	SVM	RF	K-NN
Messina	R^2	0.9595	0.9497	0.9636	0.6463
	MSE	1.5723	1.9517	1.4138	13.7267
	RMSE	1.2539	1.3970	1.1890	3.7050
Mutale	R^2	0.9898	0.8107	0.9902	0.6493
	MSE	0.4245	7.8700	0.4085	14.5817
	RMSE	0.6515	2.8053	0.6391	3.8186

Table 4.2, shows that RF model perfomed better than any other model on every station, this can be seen by the model having larger R^2 and smaller MSE and RMSE. The model was then followed by XGB, SVM, and lastly K-NN.

Chapter 5

Conclusions and future directions

5.1 Introduction

This chapter summarizes the research findings and proposes some recommendations. The study's limitations, including areas for future research, are also discussed in this chapter.

5.2 Study findings

This project explored the analysis of solar radiation. All the findings for this project are based on the GSR time-series data for two sites in South Africa; Mutale (from 2006 to 2020), Messina (from 2004 to 2020) stations retrieved from https://sauran.ac.za//. The main reason for the selected period was that it covers the recent and old timeframes, and all the seasons.

An overall of 4 models, namely RF, XGBoost, SVM, and K-NN, were used to forecast the signals of GRS in which the RF model on Mutale station was found to be the best fitting model with $R^2 = 0.9902$, $MSE = 0.4085$ and $RMSE = 0.6391$, followed by XGB with $R^2 = 0.9898$, $MSE = 0.4245$ and $RMSE = 0.6515$.

The best model for forecasting GSR from Messina station is RF with $R^2 = 0.9636$, $MSE = 0.1.4138$ and $RMSE = 1.1890$, followed by XGB model with $R^2 = 0.9595$, $MSE = 1.5723$ and $RMSE = 1.2539$.

5.3 Limitations

The limitation of this study is that only four type models in number of models were used and the comparison was made in four models.

5.4 Conclusion

Mutale station

The results from Table 4.2, supported by Figures 4.3 (A)-(D) shows that the best model to forecast Mutale GSR is the Random Forest. From the Figures 4.3 (C) and 4.3 (A), the blue line is much closer to the red line(observed value). This implies that the Random Forest model is the best model to predict and forecast GSR for Mutale station.

Messina station

The results from Table 4.2, supported by Figures 4.3 (A)-(D) shows that the best model to forecast Mutale GSR is the Random Forest. From the Figures 4.3 (C) and 4.3 (A), the blue line(predicted value) is much closer to the red line(observed value). This implies that the Random Forest model is the best model to predict and forecast GSR for Messina station.

5.5 Areas of future study

Future research should look at forecasting GSR signal using a combination of several statistical, mathematical and Machine learning models. Also, future studies should look at modelling GSR for several sites to see which model is the best based on having the smallest RMSE, MSE, and bigger R^2 on both or most stations. This will help check for the persistence of volatility in the renewable energy sector.

References

[1] A Angstrom. "Computation of global radiation from records of sunshine". In: *Ark. Geofys.;(Sweden)* 2 (1956).

[2] Slavica Brkić and Blanka Tuka. "Empirical Model for Estimating Solar Radiation Based on Air Temperature for Sarajevo Area, Bosnia and Herzegovina". In: 8 (Dec. 2019), pp. 13–29.

[3] M Chegaar, A Lamri, and A Chibani. "Estimating global solar radiation using sunshine hours". In: ().

[4] Hamid Ettayyebi and Khalid El Himdi. "Artificial neural networks for forecasting the 24 hours ahead of global solar irradiance". In: *AIP Conference Proceedings*. Vol. 2056. 1. AIP Publishing LLC. 2018, p. 020010.

[5] EO Falayi and AB Rabiu. "Solar radiation models and information for renewable energy applications". In: *Solar radiation, EB Babatunde, IntechOpen* (2012), pp. 111–130.

[6] Tamara Rosemary Govindasamy and Naven Chetty. "Machine learning models to quantify the influence of PM10 aerosol concentration on global solar radiation prediction in South Africa". In: *Cleaner Engineering and Technology* 2 (2021), p. 100042.

[7] Mawloud Guermoui et al. "Daily global solar radiation modelling using multi-layer perceptron neural networks in semi-arid region". In: *Leonardo electronic journal of practices and technologies* 28 (2016), pp. 35–46.

[8] Chien-An Hu et al. "Using a machine learning approach to predict mortality in critically ill influenza patients: a cross-sectional retrospective multicentre study in Taiwan". In: *BMJ open* 10.2 (2020).

[9] PC Jain. "Global irradiation estimation for Italian locations". In: *Solar & wind technology* 3.4 (1986), pp. 323–328.

[10] Eric N Maluta, Tshimangadzo S Mulaudzi, and Vaith Sankaran. "Estimation of the global solar radiation on the horizontal surface from temperature data for the Vhembe District in the Limpopo Province of South Africa". In: *International journal of green energy* 11.5 (2014), pp. 454–464.

[11] A Marzo et al. "Daily global solar radiation estimation in desert areas using daily extreme temperatures and extraterrestrial radiation". In: *Renewable Energy* 113 (2017), pp. 303–311.

[12] Seyed Abbas Mousavi Maleki, H Hizam, and Chandima Gomes. "Estimation of hourly, daily and monthly global solar radiation on inclined surfaces: Models re-visited". In: *Energies* 10.1 (2017), p. 134.

[13] Sophie T Mulaudzi, Vaithianathaswami Sankaran, and Meena D Lysko. "Solar radiation analysis and regression coefficients for the Vhembe Region, Limpopo Province, South Africa". In: *Journal of Energy in Southern Africa* 24.3 (2013), pp. 02–07.

[14] Ngoc Thanh Nguyen et al. "Computational Collective Intelligence-11th International Conference, ICCCI 2019, Proceedings, Part II". In: *1th International Conference, ICCCI 2019*. 2019, 702p.

[15] JA Prescott. "Evaporation from a water surface in relation to solar radiation". In: *Trans. Roy. Soc. S. Aust.* 46 (1940), pp. 114–118.

[16] John W Tukey. *Exploratory data analysis*. Vol. 2. Reading, MA, 1977.

[17] Suwan Wu, Li Jia, and Yining Liu. "Ultra-short-term wind energy prediction based on wavelet denoising and multivariate LSTM". In: *2021 Power System and Green Energy Conference (PSGEC)*. IEEE. 2021, pp. 443–447.

Appendix A

Codes for Data Analysis

A.1 Codes

The below mini page shows a link for codes that was used for the whole research.

https://drive.google.com/drive/folders/14sJrBupy1v0yvWm-QEVXww_-_U5gt8Re?usp=drive_link//

Appendix The folder contains two pdf files.

Prediction of Global Solar Radiation in Selected Areas of Limpopo Province using Selected Emperical and Machine Learning Models.

Prepared by Mulalo Marandela

Load the libraries required to analyse the data

In [164]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\2380323\anaconda3\new fold
er\lib\site-packages (1.5.0)
Requirement already satisfied: scipy in c:\users\2380323\anaconda3\new folder
\lib\site-packages (from xgboost) (1.6.2)
Requirement already satisfied: numpy in c:\users\2380323\anaconda3\new folder
\lib\site-packages (from xgboost) (1.20.1)
```

In [165]:
```
# Data manipulation and visualisation libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from math import sqrt
import datetime
import time

# Data prepocessing libraries
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split

# Model perfomance measures and hyper-parameter tunning libraries
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV, RepeatedC

# Machine Learning Libraries
from sklearn.svm import SVR
from sklearn import neighbors
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from xgboost import XGBRegressor, XGBRFRegressor

%matplotlib inline
font = {
    'family': 'DejaVu Sans',
    'weight': 'bold',
    'size'   : 20}
plt.rc('font',**font)
```

Load the data

In [166]:
```
Data = pd.read_excel('ARC Data.xlsx',sheet_name='Messina',skiprows=5)
Data.head()
```

Out[166]:

	Compno	Year	Month	Day	Tx	Tn	RHx	RHn	Rs	U2	Rain	ET0
0	30585	2003	2	1	--	--	--	--	29.6	2.05	0	6.93
1	30585	2003	2	2	--	--	--	--	29.43	1.38	0	7.24
2	30585	2003	2	3	--	--	--	--	28.65	1.9	0	7.09
3	30585	2003	2	4	--	--	--	--	29.93	1.83	0	7
4	30585	2003	2	5	--	--	--	--	28.8	1.34	0	7.12

Visualise the data

In [172]:
```
#plt.style.use('classic')
plt.figure(figsize=(8,5))
sns.set(style = "darkgrid")
sns.color_palette("viridis", as_cmap=True)
ax = sns.lineplot(data=New_df, x="Date", y="Tx")
ax.set(xlabel='Period', ylabel='Maximum Tempatarture')
plt.title("Maximum Tempatarture ")
ax.grid(True)
plt.show()
```

In [173]:
```
#plt.style.use('classic')
plt.figure(figsize=(8,5))
sns.set(style = "darkgrid")
sns.color_palette("viridis", as_cmap=True)
ax = sns.lineplot(data=New_df, x="Date", y="Tn")
ax.set(xlabel='Period', ylabel='Minimum Tempatarture')
plt.title("Minimum Tempatarture ")
ax.grid(True)
plt.show()
```

In [174]:
```
#plt.style.use('classic')
plt.figure(figsize=(8,5))
sns.set(style = "darkgrid")
sns.color_palette("viridis", as_cmap=True)
ax = sns.lineplot(data=New_df, x="Date", y="RHx")
ax.set(xlabel='Period', ylabel='RHx')
#plt.title("Minimum Tempatarture ")
ax.grid(True)
plt.show()
```

In [178]:

```
#plt.style.use('classic')
plt.figure(figsize=(8,5))
sns.set(style = "darkgrid")
sns.color_palette("viridis", as_cmap=True)
ax = sns.lineplot(data=New_df, x="Date", y="ET0")
ax.set(xlabel='Period', ylabel='ET0')
#plt.title("Minimum Tempatarture ")
ax.grid(True)
plt.show()
```

In [167]:
```
Data.tail()
```

Out[167]:

	Compno	Year	Month	Day	Tx	Tn	RHx	RHn	Rs	U2	Rain	ET0
6943	Rain	Total Daily Rainfall	mm	MWS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6944	APan	Total Daily Apan Evaporation	mm	MWS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6945	UTot	Daily Wind Run	KM/day	MWS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6946	Suns	Sunshine Hours	Hours	MWS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6947	AveT	Daily Average Temperature [[Tx + Tn] / 2]	°C	MWS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Clean the data

In [168]:
```
def CleanData(df):

    # Get names of indexes for which column "Compno" has value Compno
    indexNames = df[df['Compno'] == 'Compno'].index

    # Delete these row indexes from dataFrame
    df.drop(indexNames, inplace=True)

    # From the Day column, delete rows with the following entries Average, Tot
    df = df[df.Day != 'Average']
    df = df[df.Day != 'Total']
    df = df[df.Day != 'Highest']
    df = df[df.Day != 'Lowest']
    df = df.dropna(how= 'all')
    df = df.replace('--',np.nan)
    df = df.dropna(subset=['Tx', 'Tn', 'RHx', 'RHn', 'Rs', 'U2','Rain', 'ET0']
    dates = [str(int(day)) + '-' + str(int(month)) + '-' + str(int(year)) for
    dates = [datetime.datetime.strptime(date, '%d-%m-%Y') for date in dates]
    df['Date'] = dates
    df['Temp_diff'] = df['Tx'] - df['Tn']
    df = df[['Date','Year', 'Month', 'Day', 'Tx', 'Tn', 'RHx', 'RHn', 'U2','Ra
    df = df.interpolate(method ='linear', limit_direction ='forward')
    df['Date'] = pd.to_datetime(df.Date)
    return df
```

Split the data into test and training set

In [183]:
```
Data_mod = ExtractTempModelingFeatures(New_df)
Data_mod = Data_mod.drop('Date',axis=1)
Data_mod.head()
```

Out[183]:

	Year	Month	Day	Tx	Tn	RHx	RHn	U2	Rain	ET0	Rs
17	2003	2	18	36.1	24.5	83.7	28.1	2.38	0.0	6.04	23.95
18	2003	2	19	39.5	23.0	69.4	23.3	1.72	0.0	6.40	26.09
19	2003	2	20	39.7	22.8	93.0	25.6	1.57	14.4	4.00	16.49
20	2003	2	21	31.0	23.2	93.4	58.0	1.66	0.4	3.14	14.13
21	2003	2	22	33.0	21.0	85.6	35.9	1.74	0.3	3.87	15.67

In [184]:
```
Features = Data_mod.drop('Rs', axis = 1)
Target = Data_mod['Rs']
Features_list = list(Data.columns)
#Features = Data.values
X_train, X_test, y_train, y_test = train_test_split(Features, Target,test_size
                                                    random_state = 42,shuffle=

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)
```

```
Training Features Shape: (3868, 10)
Training Labels Shape: (3868,)
Testing Features Shape: (968, 10)
Testing Labels Shape: (968,)
```

In [185]:
```
X_train.to_csv('TrainCalibration.csv',index=False)
```

Scale the data

In [186]:
```
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(X_train)
x_train = pd.DataFrame(x_train_scaled)

x_test_scaled = scaler.fit_transform(X_test)
x_test = pd.DataFrame(x_test_scaled)
```

Application of Machine Learning Models

The k Nearest Neighbours Algorithm

In [187]:
```
def KnnMethod(K_opt,x_train,x_test,y_train,y_test):
    '''
    Apply the kNN method with optimal K

    '''
    model = neighbors.KNeighborsRegressor(**K_opt)
    model.fit(x_train, y_train)  #fit the model
    pred = model.predict(x_test) #make prediction on test set
    error = sqrt(mean_squared_error(y_test,pred)) #calculate rmse
    return pred
```

Perfom the hyper-parameter tuning to find the optimal value of k

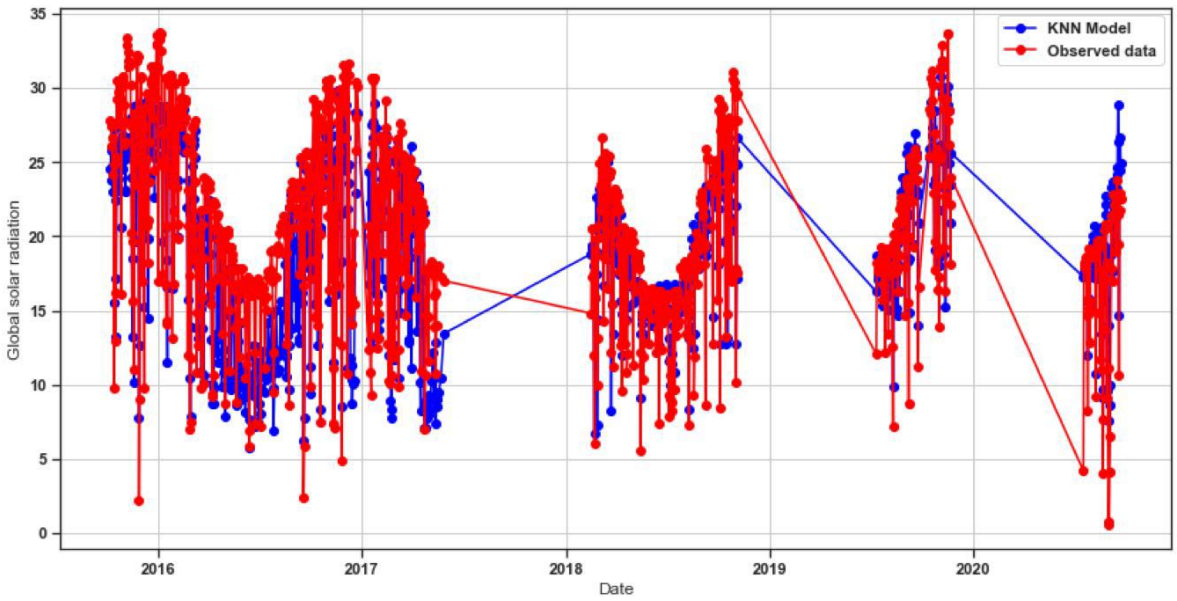
In [188]:
```
def GridSearchKNN(K_vals,x_train,x_test):

    params = {'n_neighbors':K_vals}
    knn = neighbors.KNeighborsRegressor()
    model = GridSearchCV(knn, params, cv=3)
    model.fit(x_train,y_train)

    return model.best_params_

K = [i for i in range(2,50)]
bst_k = GridSearchKNN(K,x_train,x_test)
bst_k
```

Out[188]:
```
{'n_neighbors': 7}
```

Make predictions using the Support Vector Regression with Optimal Parameters¶

In [189]:
```
KNN = KnnMethod(bst_k,x_train,x_test,y_train,y_test)
dates = [str(int(day)) + '-' + str(int(month)) + '-' + str(int(year)) for day,
dates = [datetime.datetime.strptime(date, '%d-%m-%Y') for date in dates]
```

In [190]:
```
plt.figure(figsize=(14,7))
plt.plot(dates, KNN,'ro-',label='KNN Model',color="blue")
plt.plot(dates, y_test.values,'bo-',label='Observed data',color="red")
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



Application of Support Vector Regression

In [191]:
```
def SVRMethod(x_train,x_test,y_train,params):

    SVReg = SVR(**params)
    SVReg.fit(x_train,y_train)

    #5 Predicting a new result
    pred = SVReg.predict(x_test)
    return pred
```

Perform hyperparameter tuning to obtain the optimal parameters to use for the SVR

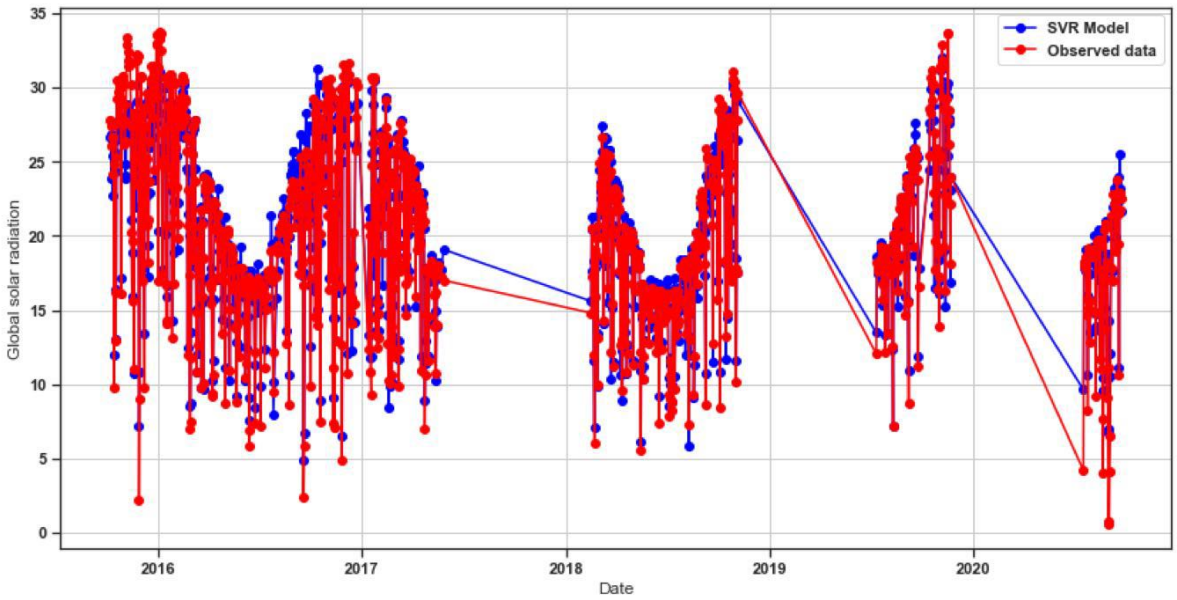
In [192]:
```
model_svr = SVR()
kernel = ["linear", "rbf", "poly"]
tolerance = [1e-3, 1e-4, 1e-5, 1e-6]
gamma = np.arange(0.01,1.0,20)
C = np.arange(1,20,20)
grid = dict(kernel=kernel, tol=tolerance, C=C)

cvFold = RepeatedKFold(n_splits=5, n_repeats=5, random_state=1)
randomSearch = RandomizedSearchCV(estimator=model_svr, n_jobs=-1,
                                  cv = cvFold, param_distributions=grid,
                                  scoring="neg_mean_squared_error")

searchResults = randomSearch.fit(x_train, y_train)

# extract the best model and evaluate it
print("[INFO] evaluating...")
bestModel = searchResults.best_estimator_
print("R2: {:.2f}".format(bestModel.score(x_test, y_test)))
searchResults.best_params_
```

```
[INFO] evaluating...
R2: 0.95
```

Out[192]:
```
{'tol': 0.001, 'kernel': 'rbf', 'C': 1}
```

Make predictions using the Support Vector Regression with Optimal Parameters

In [193]:
```
Opimal_SVM_params = searchResults.best_params_
SVR = SVRMethod(x_train,x_test,y_train,Opimal_SVM_params)
plt.figure(figsize=(14,7))
plt.plot(dates, SVR,'ro-',label='SVR Model',color="blue")
plt.plot(dates, y_test.values,'bo-',label='Observed data',color="red")
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



Application of Random Forest

In [194]:
```
def RFMethod(x_train,x_test,y_train,params):

    # Instantiate the model
    Opti_rf = RandomForestRegressor(**params)
    Opti_rf.fit(x_train,y_train)

    #5 Predicting a new result
    pred = Opti_rf.predict(x_test)
    return pred
```

Perform hyperparameter tuning to obtain the optimal parameters to use for the Random Forest Algorithm

In [195]:
```
def RandomizedParamSearch(X_train, y_train):

    # First create the base model to tune
    #rf = GetBaseModelParams(test_features, test_labels)
    rf = RandomForestRegressor()

    # Number of trees in random forest
    n_estimators = [int(x) for x in np.linspace(start = 50, stop = 500, num =

    # Number of features to consider at every split
    max_features = ['auto', 'sqrt']

    # Maximum number of levels in tree
    max_depth = [int(x) for x in np.linspace(10, 100, num = 10)]
    max_depth.append(None)

    # Minimum number of samples required to split a node
    min_samples_split = [2, 5, 10]

    # Minimum number of samples required at each leaf node
    min_samples_leaf = [1, 2, 4]

    # Method of selecting samples for training each tree
    bootstrap = [True, False]

    # Create the random grid
    random_grid = {'n_estimators': n_estimators,'max_features': max_features,'
                   'min_samples_split': min_samples_split,'min_samples_leaf':
                   'bootstrap': bootstrap
                  }

    # - Use the random grid to search for best hyperparameters
    # - Random search of parameters, using 3 fold cross validation, search acr
    #   and use all available cores
    rf_random = RandomizedSearchCV(estimator=rf, param_distributions=random_gr
                                   scoring='neg_mean_absolute_error',cv = 5, v
                                   random_state=42, n_jobs=-1)

    # Fit the random search model
    rf_random.fit(X_train, y_train)

    return rf_random
```

In [196]:
```
BestRF = RandomizedParamSearch(X_train,y_train)
Optimal_RF = BestRF.best_params_
Optimal_RF
```

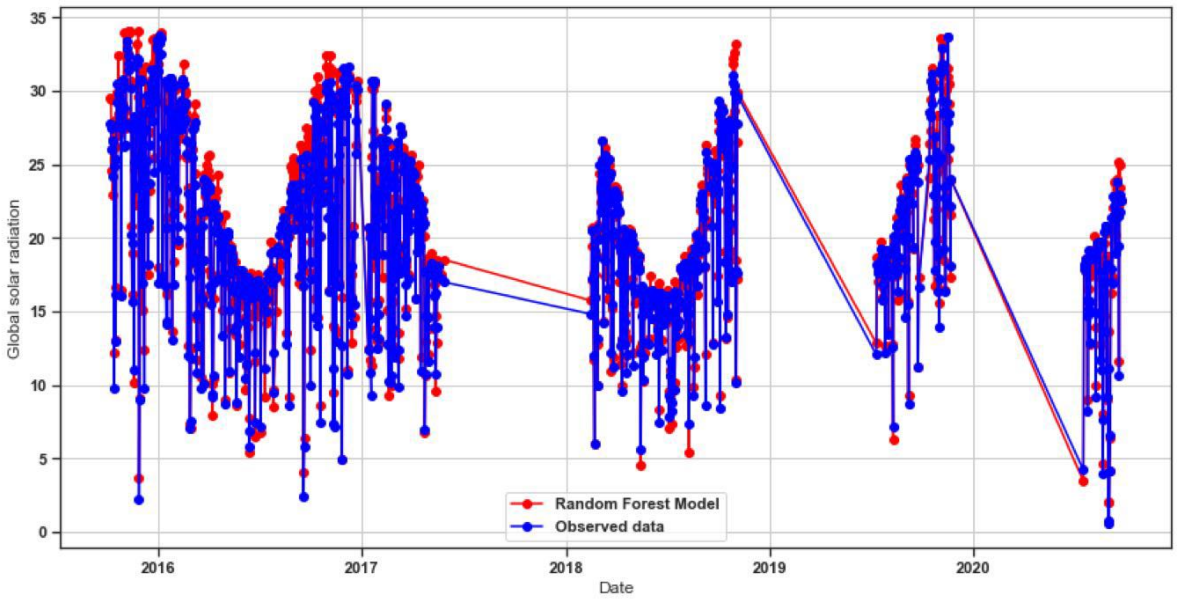
```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
```

Out[196]:
```
{'n_estimators': 300,
 'min_samples_split': 2,
 'min_samples_leaf': 2,
 'max_features': 'auto',
 'max_depth': 50,
 'bootstrap': True}
```

Make predictions using Random Forest with Optimal Parameters

In [197]:
```
RF = RFMethod(X_train,X_test,y_train,Optimal_RF)
```

In [198]:
```
plt.figure(figsize=(14,7))
plt.plot(dates, RF,'ro-',label='Random Forest Model',color="red")
plt.plot(dates, y_test.values,'bo-',label='Observed data',color="blue")
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



Application of Exteme Gradient Boosting (XGBoost) Algorithm

In [199]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\2380323\anaconda3\new fold
er\lib\site-packages (1.5.0)
Requirement already satisfied: scipy in c:\users\2380323\anaconda3\new folder
\lib\site-packages (from xgboost) (1.6.2)
Requirement already satisfied: numpy in c:\users\2380323\anaconda3\new folder
\lib\site-packages (from xgboost) (1.20.1)
```

In [200]:
```
def XGBoostMethod(x_train,x_test,y_train,params):

    # Instantiate the model
    xgb_model = XGBRegressor(**params)
    xgb_model.fit(x_train,y_train)

    # Predicting a new result
    pred = xgb_model.predict(x_test)
    return pred
```

Perform hyperparameter tuning to obtain the optimal parameters to use for the XGBoost Algorithm

In [201]:
```
#XGBoost hyper-parameter tuning
def hyperParameterTuning(X_train, y_train):
    param_tuning = {'learning_rate': [0.01, 0.1],'max_depth': [3, 5, 7, 10],
                    'min_child_weight': [1, 3, 5],'subsample': [0.5, 0.7],
                    'colsample_bytree': [0.5, 0.7],'n_estimators' : [100, 200,
                    'objective': ['reg:squarederror']
                   }

    xgb_model = XGBRegressor()

    gsearch = GridSearchCV(estimator = xgb_model, param_grid = param_tuning,
                           cv = 5,n_jobs = -1,verbose = 1)

    gsearch.fit(X_train,y_train)

    return gsearch.best_params_
```

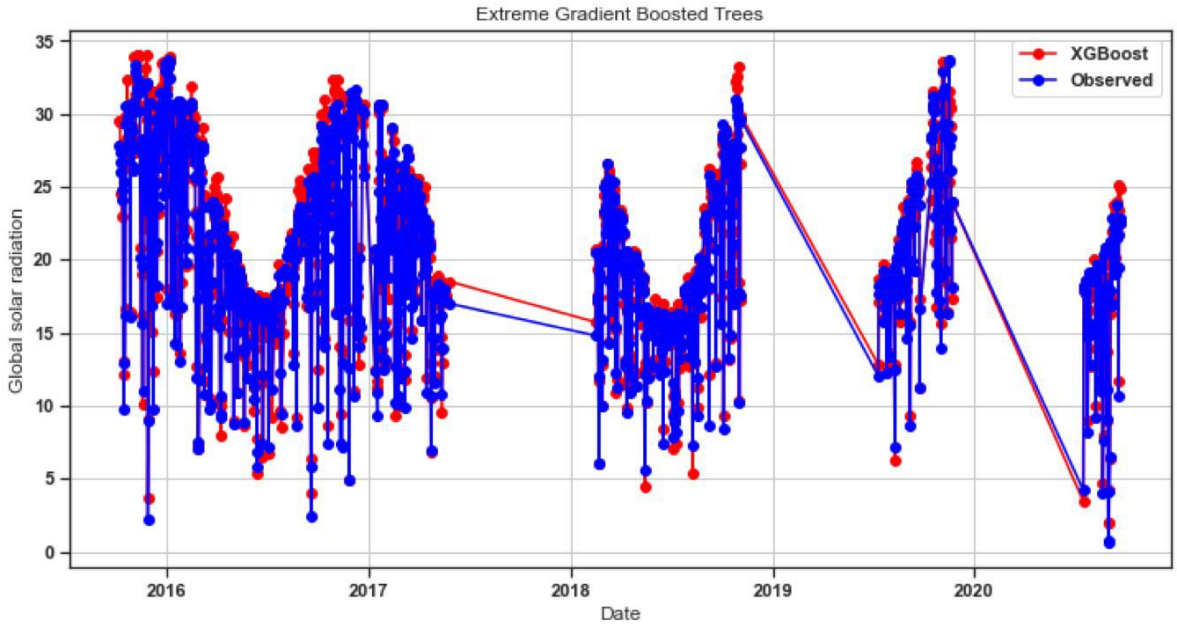
In [202]:
```
XGB_parms = hyperParameterTuning(X_train.values, y_train.values)
XGB_parms
```

```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
```

Out[202]:
```
{'colsample_bytree': 0.7,
 'learning_rate': 0.1,
 'max_depth': 3,
 'min_child_weight': 1,
 'n_estimators': 500,
 'objective': 'reg:squarederror',
 'subsample': 0.7}
```

Make predictions using the XGBoost Algorithm with Optimal Parameters

In [203]:
```
XGBoost = XGBoostMethod(X_train.values,X_test.values,y_train.values,XGB_parms)
plt.figure(figsize=(12,6))
plt.plot(dates, RF,'ro-',label='XGBoost',color="red")
plt.plot(dates, y_test.values,'bo-',label='Observed',color="blue")
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.title('Extreme Gradient Boosted Trees')
plt.legend()
plt.grid(True)
```



Application of Adaptive Gradient Boosting (AdaBoost) Algorithm

In [204]:
```
def AdaGradBoostMethod(x_train,x_test,y_train,params):

    # Instantiate the model
    AdaGrad = AdaBoostRegressor(**params)
    AdaGrad.fit(x_train,y_train)

    # Predicting a new result
    pred = AdaGrad.predict(x_test)
    return pred
```

Perform hyperparameter tuning to obtain the optimal parameters to use for the AdaBoost Algorithm

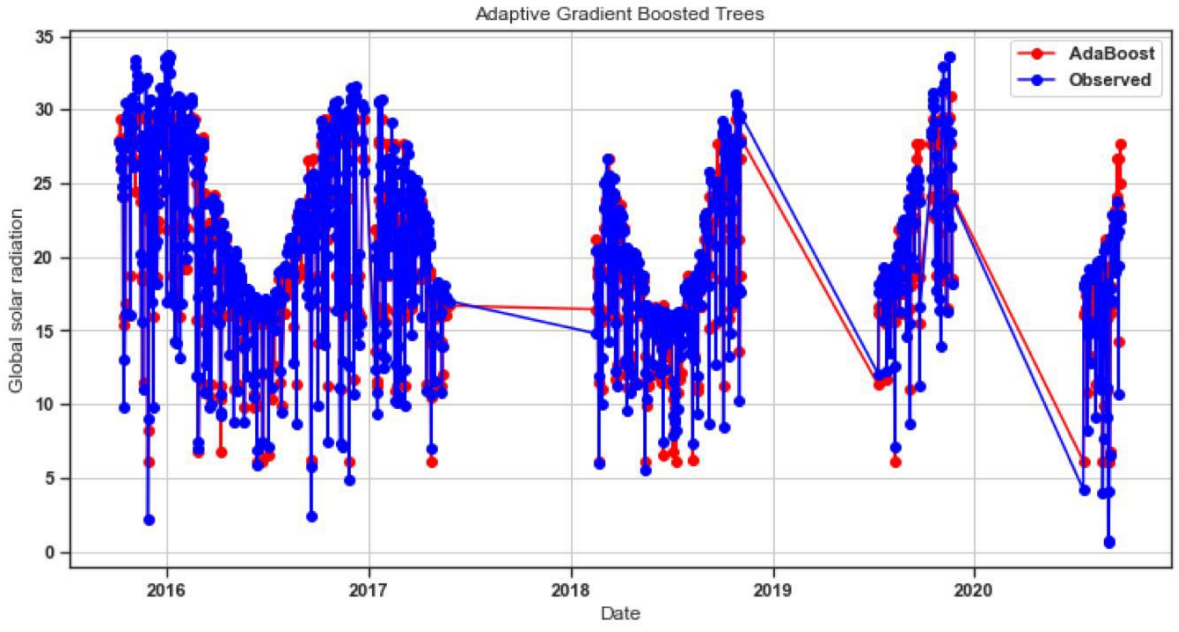
In [205]:
```
def AdaBoostTune(X_train,y_train):

    param_dist = {'n_estimators': np.arange(100,500,10),'learning_rate' : np.a
                  'loss' : ['linear', 'square', 'exponential']
                 }
    AdaGrad = AdaBoostRegressor()
    AdaParams = RandomizedSearchCV(AdaGrad, param_distributions = param_dist,
                                   cv=5, n_iter = 30, n_jobs=-1)
    AdaParams.fit(X_train, y_train)

    return AdaParams.best_params_
```

In [206]:
```
AdaBoost_optim = AdaBoostTune(X_train.values,y_train.values)
AdaBoost_optim
```

Out[206]:
```
{'n_estimators': 450, 'loss': 'exponential', 'learning_rate': 0.01}
```

Make predictions using the AdaBoost Algorithm with Optimal Parameters

In [207]:
```
AdaBoost = AdaGradBoostMethod(X_train.values,X_test.values,y_train.values,AdaB
plt.figure(figsize=(12,6))
plt.plot(dates, AdaBoost,'ro-',label='AdaBoost',color="red")
plt.plot(dates, y_test.values,'bo-',label='Observed',color="blue")
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.title('Adaptive Gradient Boosted Trees')
plt.legend()
plt.grid(True)
```



Application of the Emperical Models

The Clemence model

We use the Clemence model given by the following formula

$$H = (1.233\ H_0 \Delta T + 10.593\ T_{max} - 0.713\ T_{max}\ \Delta T + 16.5480) \cdot (0.04184)$$

where H_0 is given by the following equation

$$H_0 = \frac{24(60)}{\pi} \cdot G_{sc} \cdot d_r\ [\omega_s \sin(\varphi) \sin(\delta) + \cos(\varphi) \cos(\delta) \sin(\omega_s)]$$

with $d_r = 1 + 0.033 \cos\left(\frac{2\pi}{365} J\right)$ and $\delta = 0.409 \sin\left(\frac{2\pi}{365} J - 1.39\right)$. Here J is the number of the day in the year between 1 (1 January) and 365 or 366 (31 December). The variable J can be determined for each day (D) of month (M) by using the following recursive algorithm:

$$J = \textbf{INTEGER}\left(275 \times \frac{M}{9} - 30 + D\right) - 2.$$

If $M < 3$, then $J = J + 2$. Also if it is a leap year and $M > 2$, then $J = J + 1$.

The sunset hour angle, ω_s is defined as:

$$\omega_s = \cos^{-1}[-\tan(\phi) \tan(\delta)].$$

Define the parameters to use

In [208]:
```
#Importing msth

import math
```

In [209]:
```
Tx = X_test['Tx'].values   # Maximum Temp
Tn = X_test['Tn'].values   # Maximum Temp
Dx = Tx - Tn # Temparature difference
phi = (np.pi/180)*(-22.73461)
M = X_test.Month.values    # Months column
D = X_test.Day.values      # Days column
Y = X_test.Year.values     # Year column
GSC = 0.0820
```

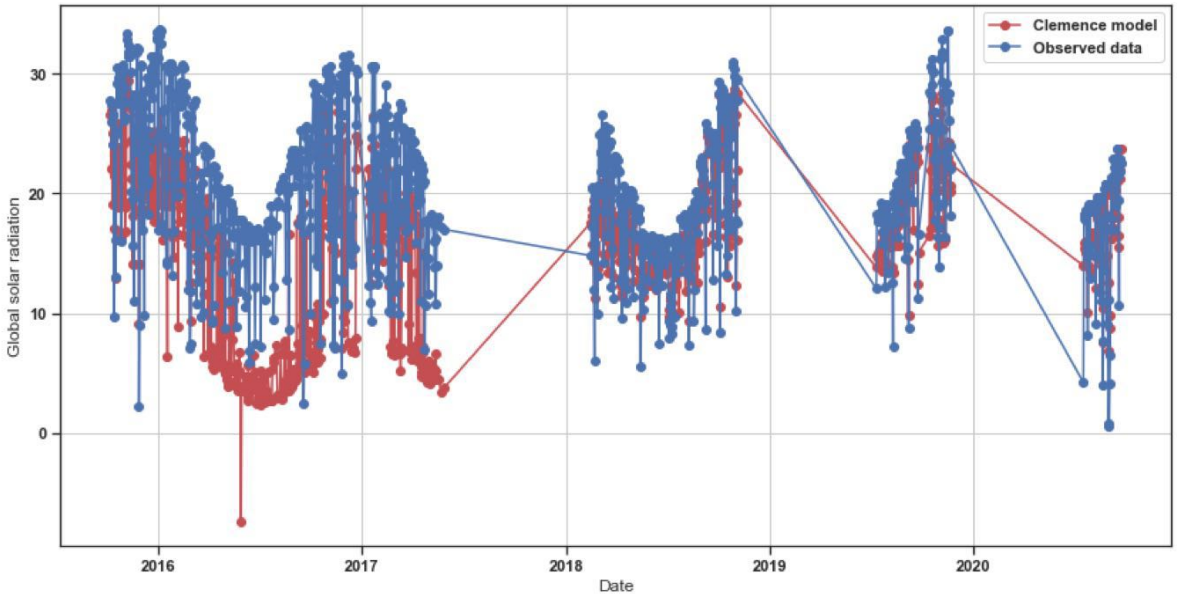
In [210]:
```
# Function for cheking if the year is classified as leap year
def IsLeapYear(Year):
    if (Year % 4) == 0:
        if (Year % 100) == 0:
            if (Year % 400) == 0:
                return True
            else:
                return False
        else:
             return True
    else:
        return False

# Function to claculate the variable J in the equation
def DaysofYear(Day,Month,Year):
    J = []
    for i in range(len(Day)):
        J_vals = int((275*Month[i]/9 - 30 + Day[i]) - 2)
        if Month[i] < 3:
            J_vals = J_vals + 2

            if (IsLeapYear(Year[i])== True and (Month[i]>2)):
                J_vals = J_vals + 1

            else:
                pass

        J.append(J_vals)

    return J

# Function to calculate solar declination
def Delta(Dn):
    delta = []
    for i in range(len(Dn)):
        delta.append(0.409e0 * math.sin(0.2e1 / 0.365e3 * math.pi * Dn[i] - 0.
        #delta.append(0.2345e2 * math.sin(0.20448e5 / 0.73e2 * Dn[i]))
    return delta

# Function to calculate the inverse relative distance Earth-Sun
def Earth_Sun(Dn):
    dr = []
    for i in range(len(Dn)):
        dr.append(1 + 0.33e-1 * math.cos(0.2e1 / 0.365e3 * math.pi * Dn[i]))

    return dr

# Function to calculate sunset hour angle
def SunsetHour(Dn,delta,phi):
    delta = Delta(Dn)
    Ws = []
    for i in range(len(Dn)):
        Ws.append(math.acos(-math.tan(phi) * math.tan(delta[i])))

    return Ws

# Function to calculate the daily extraterrestrial solar radiation
```

```
def ExtraTerestialRadiation(Day,Month,phi,Year,GSC = 0.0820):
    dn = DaysofYear(Day,Month,Year)
    dr = Earth_Sun(dn)
    delta = Delta(dn)
    Ws = SunsetHour(dn,delta,phi)
    Ra = []
    for i in range(len(Day)):
        H0 = 1440 / math.pi * GSC * dr[i] * (Ws[i] * math.sin(phi) * math.sin(
                                            + math.cos(phi) * math.cos(delta[i
        Ra.append(H0)

    return Ra

H0 = ExtraTerestialRadiation(D,M,phi,Y)
```

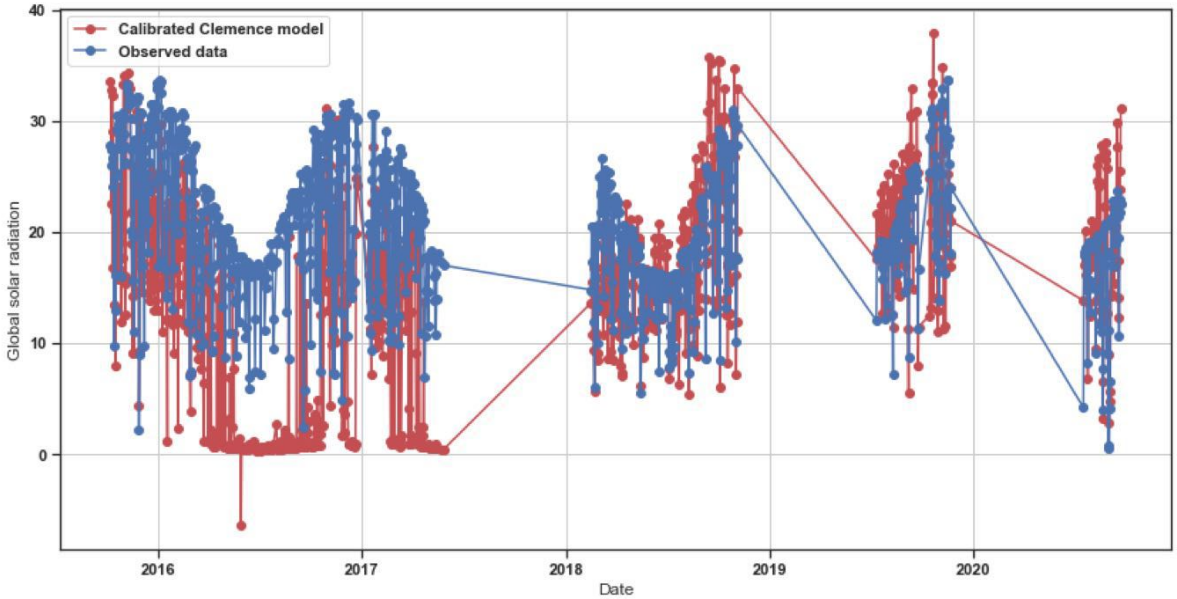
In [211]:
```
# Function to implement the clemence model
def ClemenceModel(Tx,Dx,H0):
    H_vals = []
    for i in range(0,len(Tx)):
        H = 0.03184*(1.233*H0[i]*Dx[i] + 8.593*Tx[i] - 0.713*Tx[i]*Dx[i] + 16.
        H_vals.append(H)

    return H_vals

Clemence = ClemenceModel(Tx,Dx,H0)
```

In [212]:
```
# Function to implement the clemence model
def ClemenceModelCalib(Tx,Dx,H0):
    H_vals = []
    for i in range(0,len(Tx)):
        H =  (1.0/60)*(3.051)*(0.6355*H0[i]*Dx[i] + 0.4753*Tx[i] - 0.0005308*T
        #H = H/60
        H_vals.append(H)

    return H_vals
Clemence2 = ClemenceModelCalib(Tx,Dx,H0)
```

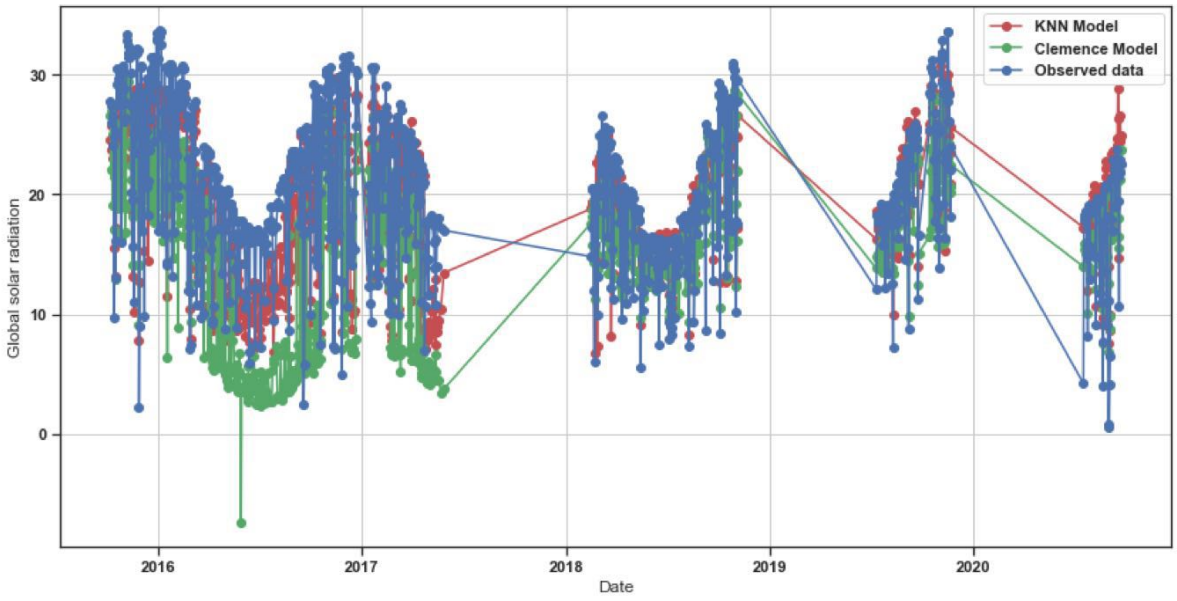
In [213]:
```
plt.figure(figsize=(14,7))
plt.plot(dates, Clemence,'ro-',label='Clemence model')
plt.plot(dates, y_test.values,'bo-',label='Observed data')
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



In [214]:
```
plt.figure(figsize=(14,7))
plt.plot(dates, Clemence2,'ro-',label='Calibrated Clemence model')
plt.plot(dates, y_test.values,'bo-',label='Observed data')
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



In [215]:
```
plt.figure(figsize=(14,7))
plt.plot(dates, KNN,'ro-',label='KNN Model')
plt.plot(dates, Clemence,'go-',label='Clemence Model')
plt.plot(dates, y_test.values,'bo-',label='Observed data')
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



The Hargreaves and Samani model

Hargreaves and Samani model used the maximum daily temperature and minimum daily temperature to estimate daily global solar radiation from using the equation below:

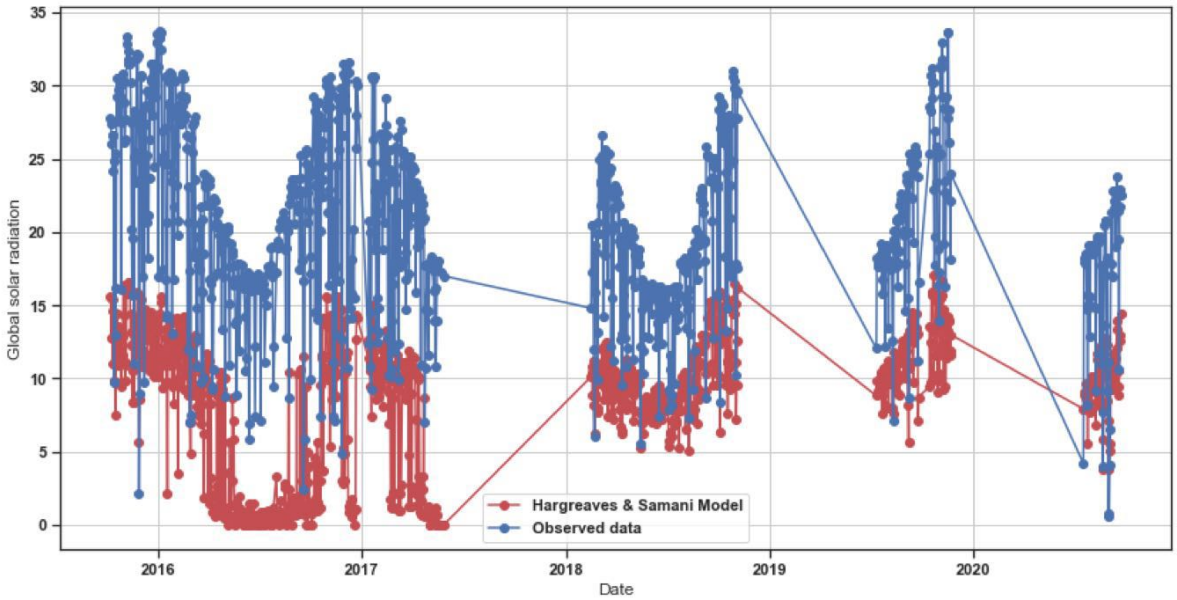
$$H = H_0 \times K_r \times \sqrt{(\Delta T)}$$

where K_r is the empirical coefficient for inland regions with the value of 0.16 and ΔT is the difference between the maximum and the minimum average daily temperature.

In [216]:
```
# Function to estimate solar radiation using Hargreave & Samani Model
def EmpiricalCoefficient(Dx):
    K = []
    for i in range(len(Dx)):
        K.append(0.00185*(Dx[i])**2 - 0.0433*Dx[i] + 0.04023)
    return K

def HagreeavesSamani(Dx,H0):
    H_vals = []
    Kr = np.abs(np.mean(EmpiricalCoefficient(Dx)))
    print('The value of Kr is {}'.format(np.round(Kr,4)))
    for i in range(0,len(Tx)):
        H = H0[i]*Kr*np.sqrt(Dx[i])
        H_vals.append(H)

    return H_vals

HagreaSama = HagreeavesSamani(Dx,H0)
```

```
The value of Kr is 0.0807

<ipython-input-216-79e2fa883e19>:13: RuntimeWarning: invalid value encountere
d in sqrt
  H = H0[i]*Kr*np.sqrt(Dx[i])
```

In [217]:
```
plt.figure(figsize=(14,7))
plt.plot(dates, HagreaSama,'ro-',label='Hargreaves & Samani Model')
plt.plot(dates, y_test.values,'bo-',label='Observed data')
plt.xlabel('Date')
plt.ylabel('Global solar radiation')
plt.legend()
plt.grid(True)
```



In [218]:
```
pred_df2 = pd.DataFrame([dates, y_test.values, KNN, SVR, RF, XGBoost,AdaBoost,
pred_df2 = pred_df2.T
pred_df2.columns = ['Date','Observed','KNN', 'SVR','Random Forest','XGBoost','
pred_df2.head()
```

Out[218]:

	Date	Observed	KNN	SVR	Random Forest	XGBoost	AdaBoost	Clemence	Calibrate Clemenc
0	2015-10-08	27.79	24.525714	26.678639	29.519631	29.457394	27.939596	26.575208	33.52786
1	2015-10-09	27.43	25.748571	26.846916	29.402176	28.637136	29.410731	26.974208	32.79900
2	2015-10-10	26.06	23.758571	23.895329	24.59075	24.135181	26.610567	22.098714	22.59680
3	2015-10-11	26.65	24.682857	25.446473	27.875782	27.131048	27.970055	26.152863	32.26843
4	2015-10-12	24.15	22.99	22.750559	22.953987	23.294937	24.060788	19.085925	16.75655

In [219]:
```
#def mean_absolute_percentage_error(y, yhat):
    #y, yhat = np.array(y), np.array(yhat)
    #return np.mean(np.abs((y- yhat) / y)) * 100
```

In [220]:
```
#Gettind rid of infinite and  null values.

pred_df2.replace([np.inf, -np.inf], np.nan, inplace=True)
pred_df2.fillna(999, inplace=True)
```

In [221]:
```
# Function to claulate perfomance metrics
def PerfomanceMetrics(y,yhat):
    MSE = mean_squared_error(y,yhat)
    RMSE = np.sqrt(MSE)
    MBE = np.mean(y - yhat)
    MABE = np.mean(np.abs(y - yhat))
    #MAPE = mean_absolute_percentage_error(y,yhat)*100
    MAPE= np.mean(np.abs((y- yhat) / y)) * 100
    RSquare = r2_score(y,yhat)
    return MSE, RMSE, MBE, MABE, MAPE, RSquare


MSE_Cl, RMSE_Cl, MBE_Cl, MABE_Cl, MAPE_Cl, RSquare_Cl = PerfomanceMetrics(prec

MSE_ClC, RMSE_ClC, MBE_ClC, MABE_ClC, MAPE_ClC, RSquare_ClC = PerfomanceMetric

MSE_HS, RMSE_HS, MBE_HS, MABE_HS, MAPE_HS, RSquare_HS = PerfomanceMetrics(prec

MSE_KNN, RMSE_KNN, MBE_KNN, MABE_KNN, MAPE_KNN, RSquare_KNN = PerfomanceMetric

MSE_RF, RMSE_RF, MBE_RF, MABE_RF, MAPE_RF, RSquare_RF = PerfomanceMetrics(prec

MSE_SVR, RMSE_SVR, MBE_SVR, MABE_SVR, MAPE_SVR, RSquare_SVR = PerfomanceMetric

MSE_XG, RMSE_XG, MBE_XG, MABE_XG, MAPE_XG, RSquare_XG = PerfomanceMetrics(prec

MSE_AD, RMSE_AD, MBE_AD, MABE_AD, MAPE_AD, RSquare_AD = PerfomanceMetrics(prec
```

In [222]:
```
Metrics = pd.DataFrame({'Model': ['Hargreaves-Samani Model','Clemence Model','
                                  'Random Forest Model','Extreme Gradient Boos
                        'MSE': [round(MSE_HS, 4),round(MSE_Cl, 4),round(MSE_Cl
                        'RMSE': [round(RMSE_HS, 4),round(RMSE_Cl, 4),round(RMS
                        'MBE' : [round(MBE_HS, 4),round(MBE_Cl, 4),round(MBE_C
                        'MABE' : [round(MABE_HS, 4),round(MABE_Cl, 4),round(MA
                        'MAPE' : [round(MAPE_HS, 4),round(MAPE_Cl, 4),round(MA
                        'R-Squared' : [round(RSquare_HS, 4),round(RSquare_Cl,
                       })
```

In [223]:
```
Metrics
```

Out[223]:

	Model	MSE	RMSE	MBE	MABE	MAPE	R-Squared
0	Hargreaves-Samani Model	1164.5285	34.1252	10.7003	12.7849	64.3401	-29.0110
1	Clemence Model	58.1047	7.6226	4.8137	5.6357	31.6197	-0.4974
2	Calibrated Clemence Model	100.0898	10.0045	5.9711	7.4313	39.0972	-1.5794
3	K-Nearest Neighbour Model	13.7267	3.7050	1.2180	2.7867	18.7716	0.6463
4	Support Vector Regression Model	1.9517	1.3970	-0.0056	1.0051	8.0151	0.9497
5	Random Forest Model	1.4138	1.1890	-0.2545	0.8459	5.0677	0.9636
6	Extreme Gradient Boosted Model	1.5723	1.2539	-0.3664	0.8810	5.3866	0.9595
7	Adaptive Gradient Boosting Model	3.4739	1.8638	0.1146	1.4331	10.0083	0.9105

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: