Applying Machine Learning Models for Predicting Household Solar Power Production: A Comparative Analysis and Evaluation

EAEE 9305.zx2407.report
Zihao Xiao zx2407
Columbia University

ABSTRACT
In this project, we propose machine learning (ML) strategies using time series models (LSTM, Prophet), ensemble learning models (XGBooster, Random Forest), and a Neural Network model to predict daily energy generation. Utilizing real open-source data from Sydney, Australia, we scrutinize the efficacy of these models, questioning if time series prediction ML models outperform other types in predicting household solar energy generation. The study also contrasts the performance of these models with their previous work predicting large-scale solar production in California, seeking to ascertain their viability in forecasting small-scale solar energy production. We examine the reasons behind the differing performances of these models, aiming to provide a comprehensive understanding of their utility in this vital area of renewable energy.

1. Introduction

Due to the need for sustainable development, many countries have started to develop clean energy technologies. According to the IEA, sustainable energy sources such as wind and solar account for more than a quarter of the world's electricity production in 2021.**[1]**

Solar energy is an important part of clean energy and is currently the main focus. In terms of production scale, there are two types of solar energy production today, which are household solar production systems and large-scale solar power plants. Compared with solar power plants, household solar production systems allow for decentralized energy production, wich can reduce the strain on the power grid and help prevent blackouts. Besides, by generating their own power, households can achieve a level of energy independence, reducing their reliance on utility companies. For these reasons, home solar production equipment is becoming popular. And this has created a need to forecast household solar power production—for example, energy planning and grid demand balancing through forecasting.

As the rapid development of artificial intelligence, machine learning-based prediction models have been reported for various energy-related production forecasts, especially in electricity production area. Compared with various conventional methods for time series forecasting, the machine learning models has shown promising results in recent years with reduced complexity. Most of the solar energy generation prediction focus on solar radiation prediction, but it is actually have a connection with teme series.**[2]**

So in this study, we propose an ML strategy to predict household solar production. We build 2 kinds of time series ML models(LSTM, Prophet), 2 kinds of ensemble learning models(XGBooster, Random Forest) and a NN(Neural Network) model. We train these models and predict daily energy generation based on a real open source data on household energy generation in Sydney, Australia. We want to ask few questions in through this study: (1) Would time series prediction ML models provide better prediction than other models on household solar energy generation? (2) Since we also constructed XGBoost, RF and NN models to predict large-scale solar production in California last semester. So we also used the same model structure to compare and investigate whether these models still perform well in predicting small-scale solar energy production for household use in Sydney. (3) Try to analyze the reasons for the difference in performance of different models.

In the following parts, we will introduce some basic model principles and analytical methods. Next, the experimental procedure and results are shown. Finally we will analyze, summarize this work and outline future works.

2. Methodology

In this study, we used a machine learning regression algorithm. We used date and weather data(humidity, temperature, wind speed and pressure) as input. Output is the daily solar yield of 30 users. The main methodologies designed in this paper are time series ML models (Prophet and LSTM) and ensemble learning models(Random Forest and XGBoost). So in the following section, we will have a short description of the principles of these models. The specific code implementation is described later.

(1) **Prophet** is a forecasting tool developed by Facebook's Core Data Science team. Prophet is designed to handle time series data that exhibits patterns on different time scales such as yearly, weekly, and daily. It also gracefully handles holidays, missing data, and outliers. The underlying model of Prophet is essentially an additive regression model with four main components: trend, seasonality, holiday effect, and an error term.
Trend: Prophet automatically detects changes in trends by selecting changepoints from the data. It can handle both linear and logistic growth trends; Seasonality: Prophet uses a Fourier series to model seasonality. It can model multi-seasonality data by using different Fourier series for different seasonal components such as daily, weekly, and yearly seasonality; Holiday Effects: Prophet allows you to specify a list of holidays to be modeled. It creates a dummy variable for each holiday. Error Term: The error term accounts for any unusual changes not accommodated by the model.

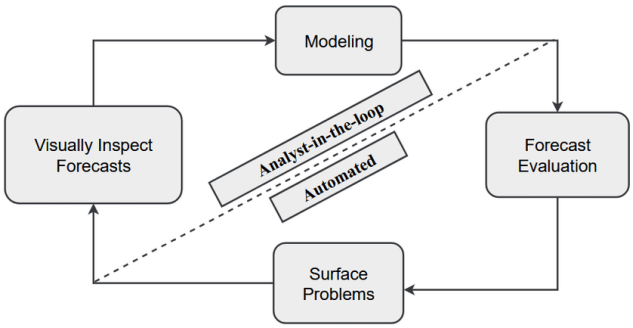


Figure 1. Flowchart of the Prophet**[3]**

Prophet uses a decomposable time series model with these components, allowing for flexible modeling of time-series data. The model is fitted via maximum a posteriori (MAP) estimation.

(2) **LSTM** (Long Short-Term Memory) is a special kind of Recurrent Neural Network (RNN) capable of learning long-term dependencies, which is especially useful in time series prediction.

The fundamental idea behind LSTM is the cell state, the horizontal line running through the top of the diagram. The cell state acts like a conveyor belt carrying information straight down the sequence chain, with only minor linear interactions. It's very easy for information to flow along it unchanged. LSTM has the ability to remove or add information to the cell state, regulated by structures called gates. Gates are composed out of a sigmoid activation layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.

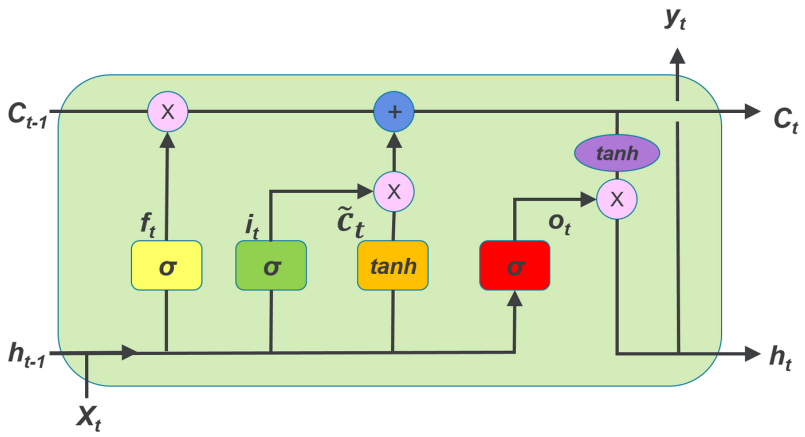


Figure 2. Flowchart of the LSTM**[4]**

(a) Forget Gate: Decides what information we're going to throw away from the cell state; (b) Input Gate: Decides which values we'll update, and a tanh layer creates a vector of new candidate values; (c) Output Gate: Decides what we're going to output based on input and the current cell state. The combination of these elements allows LSTMs to keep or forget information over long sequences, making them highly effective for time-series prediction tasks.

(3) **XGBoost** stands for eXtreme Gradient Boosting. It's a powerful machine learning algorithm that's based on the concept of 'boosting'. Boosting refers to the ensemble learning technique of building many models sequentially, with each new model attempting to correct for the deficiencies in the previous model. XGBoost creates a strong predictive model in the form of an ensemble of weak predictive models. It utilizes a gradient boosting framework which can also support regularization to prevent overfitting, which distinguishes it from other boosting algorithms.

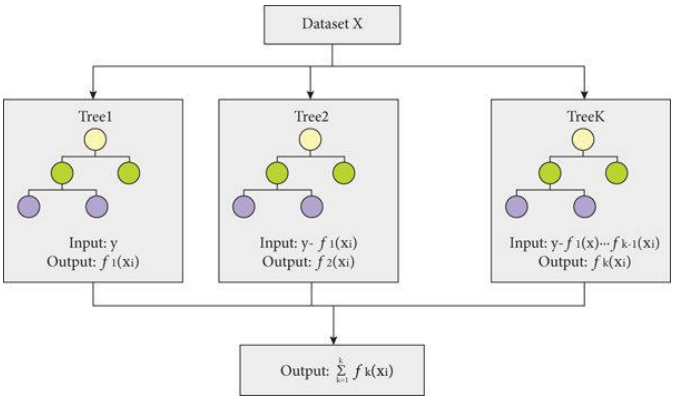


Figure 3. Flowchart of the XGBooster**[5]**

In XGBoost, trees are built sequentially, and while building these trees, the algorithm considers the learnings from the previous trees. Each new tree takes into account the errors or residuals of the previous trees, trying to correct them. This process continues until a specified number of trees are created or the error reaches an acceptable level.
XGBoost also includes a variety of regularization features that help to manage bias-variance trade-off, improving the model's performance even if the data is noisy. Moreover, it's designed to be highly efficient and flexible, with several parameters that can be tuned to optimize accuracy and computational efficiency.

(4) **Random Forest** is an ensemble machine learning algorithm that is used for both classification and regression tasks. The basic principle behind a random forest is simple but powerful: combine the predictions of several decision trees to make more accurate predictions.
A decision tree is a simple, flowchart-like structure in which internal nodes represent features (or attributes), branches represent decisions rules, and each leaf node represents an outcome. However, decision trees can be highly sensitive to the data they are trained on, often resulting in overfitting.

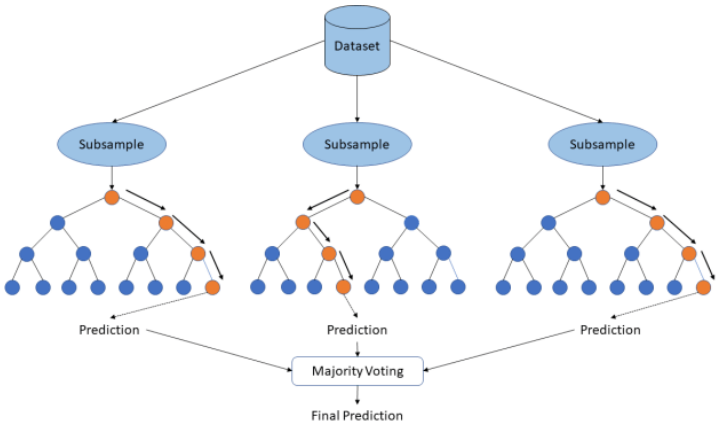


Figure 4. Flowchart of the Random Forest**[6]**

Random forest mitigates this problem by creating a whole forest of random decision trees and outputting the mode of the classes for classification or mean prediction for regression. Each decision tree in the forest is built on a subset of the data and features, introducing randomness into the model building process and resulting in diverse trees. This diversity makes the random forest model more robust and less prone to overfitting on the training data. Furthermore, random forests also have an in-built method for feature selection, giving importance scores for each feature, which can be helpful for interpretability and understanding the influential factors in the model's predictions.

In the analysis part of this article, we will use few of evaluation metrics of **regression algorithms[7]**:

The Root Mean Squared Error (RMSE) is a measure of the difference between predicted values and actual values. It is calculated as the square root of the mean squared error (MSE), which is the average of the squared differences between the predicted and actual values. Mathematically, RMSE is calculated as:

$$RMSE = sqrt(MSE) = sqrt(1/n * sum((y\_pred - y\_true)^2))$$

where y_pred is the predicted value, y_true is the actual value, and n is the number of observations.

The Residual Sum of Squares (RSS) is a measure of the total squared error between the predicted values and the actual values. It is calculated as the sum of the squared differences between the predicted and actual values. Mathematically, RSS is calculated as:

$$RSS = sum((y\_pred - y\_true)^2)$$

R-squared is a measure of how well a model fits the data. It is calculated as the proportion of the variance in the dependent variable that is explained by the model. Mathematically, R-squared is calculated as:

$$R\text{-squared} = 1 - (RSS / TSS)$$

where TSS is the total sum of squares, which is the sum of the squared differences between the actual values and the mean of the actual values.

3. Objectives and Technical Challenges

Objectives:
(1) Building time series ML models (Prophet and LSTM) and ensemble learning models(Random Forest and XGBoost) in the Anaconda environment (2) Training models and tuning parameters to improve model accuracy. (3) Comparing different models' performance, try to ask three questions mentioned before in the introduction section.

Technical Challenges:
(1) The gap between the raw and required data is large and requires more complex pre-processing. For example, it is necessary to accumulate the half-hourly capacity data for each household into daily data and match the corresponding weather data, as well as to select the user's zip code that is closer to the weather measurement station range point against the map. (2) There is limited access to weather data, especially a lack of data on solar radiation, which is most closely related to solar energy production. The Sydney region only has a weather station at Sydney Airport, resulting in the need to pick users closer to the airport. (3) An understanding of the principles of time series models and ensemble learning models, including training as well as tuning parameters is required.

4. Implementation

The flow chart below shows the progresses of this project.(Figure 5.)

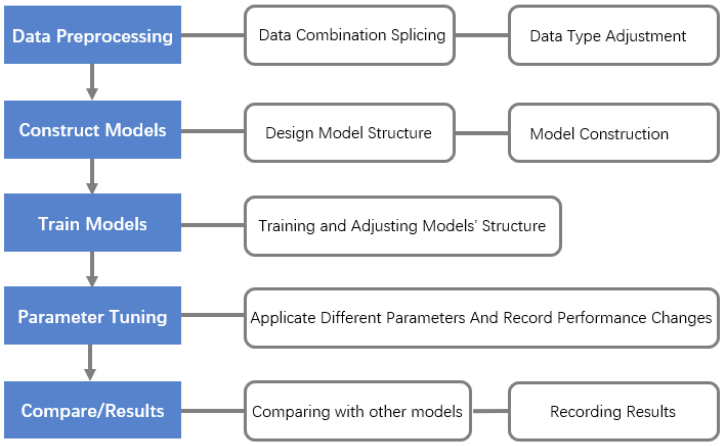


Figure 5. Flowchart of the progresses of this project.

4.1 Data and Preprocessing

We use a subset of the Solar home electricity dataset**[8]** available in Ausgrid company on the residential photovoltaic generation of 300 users selected in Sydney, Australia. The data contains users' postal code, installed PV generation capacity and solar energy generation recorded at each half hour from 2021/7/1 to 2013/6/30; We obtained weather data from wunderground**[9]**, corresponding also to the period from

2010/7/1 to 2013/6/30. We extracted four relevant data, wind speed, humidity, barometric pressure and temperature.
Since the only measurement point for this weather data is near the Kingsford Smith International Airport station in Sydney, to ensure the accuracy of the model input. We compared the zip codes of the users by map and finally selected 30 users around the airport as our final dataset.
In addition, to reduce the impact of large variance in individual users and our computational carrying capacity limitations, we summed the data from these 30 users to R as the total domestic solar power production in a small area near the airport. R can be calculated by the following formulars:

$$V(s_i, d) = \sum_{t \in d} V(s_i, t)$$

Residential photovoltaic energy generated at each day (d) each household si .And t is half hour.

$$V(d) = \sum_{i=1}^{j} \sum_{t \in d} V(s_i, t)$$

V(d) is Region-level energy generated at each day (R), j = 30.

And Fig.6 presents the values of the R value (Total airport-nearby users' energy production value) by day. We use the data from 2010 to 2012 as the training and validation set. We use the trained model to test the R value (Total airport-nearly users' energy production value) for each day in 2013. The ratio of training, validation set and test set is 65% : 35%.

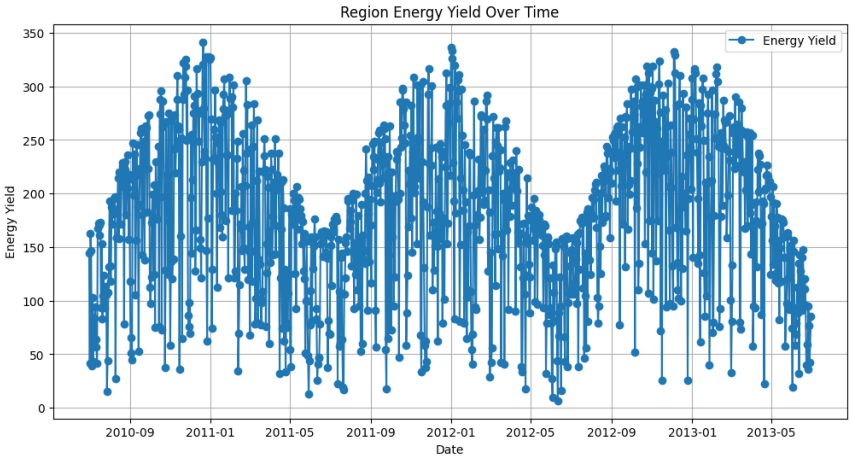


Figure 6. Region Energy Yield Over Time.

4.2 Implementation of the Machine Learning models

For our 5 different kinds of models, they all contain their own manual parameter settings. Prophet: ds(datastamp), y, changepoint, number of changepoints, prior scale of changepoint;
LSTM: Number of layers, dropout layer, batch size, learning rate, number of epoch;
XGBoost: learning rate, Minimum sum of instance weight needed in a child; Maximum depth of a tree; The number of estimators in XGBoost;
Random Forest: The number of estimators in Random Forest and random state;
Neural Network:Structures of Hidden Layers, Learning rate, Maximum itineration, activation and solver.

4.3 Software Design

Below are our software design of 6 models(The process of parameter adjustment is not included)

1. Data Prepossessing
   Algorithm:

1) Load training and testing data set.
2) Convert the Date Column to Datetime format.
3) Remove duplicate rows based on 'Date' and 'User Number' columns; remove statistical error data where the energy production is 0.
4) Sum up every users' energy generation to get R (region data).
5) Using 65:35 ratio for splitting the data, with the earlier dates being used for training and the later dates for testing.
6) Drawing the plot of Region Energy Yield Over Time and Feature Correlation Heatmap.

2. Building the model of LSTM(without using weather data), adjusting parameters and plotting graphs.
   Algorithm:
   1) Use MinMaxScaler function to scale input features.
   2) Since we do not using weather data in this model, we only remain date as input, remove other features.
   3) Create a reashape function to reshape the data in to a format suitable for the LSTM model.
   4) Set the number of previous time steps to use as input features, we set lookback = 5.
   5) Create a LSTM model. With 2 LSTM layers with 64 and 32 hidden units respectively, both using ReLu activation function; Dropout layer with a rate of 0.2; Using optimizer = Adam, learning rate = 0.001.
   6) Train the model and plot validation line chart and validation scatter plot.
   7) Calculate the corresponding RMSE, RSS and R square value.

3. Building the model of LSTM(with using weather data), adjusting parameters and plotting graphs.
   Algorithm: Same with the LSTM(without using weather data) except remain all the weather data as input.

4. Building the model of Prophet, adjusting parameters and plotting graphs.
   Algorithm:
   1) Initialize the prophet model, with chagepoint_piror_scale = 0.005, interval_width = 0.5 and changepoint_range = 0.9
   2) Add additional regressors (temperature, humidity, wind speed and pressure)
   3) Train the model and plot validation line chart and validation scatter plot.
   4) Calculate the corresponding RMSE, RSS and R square value.

5. For XGBoost, RF and NN model. Since they are not time series models, we randomly separated test and training groups. Testsize = 0.2, randomstate = 393.

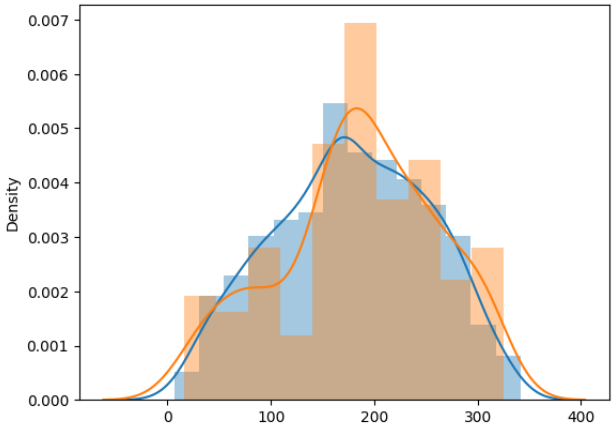


Figure 7. Distribution of the training and test sets (orange is the test set, blue is the training set)

6. Building the model of XGBoost, adjusting parameters and plotting graphs.
   Algorithm:
   1) Create a XGBooster model with estimators = 200, learning rate = 0.09, minimum child weight = 4 and maximum depth = 2.
   2) Train the model and plot validation line chart and validation scatter plot.
   3) Calculate the corresponding RMSE, RSS and R square value.

7. Building the model of Random Forest, adjusting parameters and plotting graphs.
   Algorithm:
   1) Create a XGBooster model with estimators = 700, learning rate = 0.03, random_state = 0.
   2) Train the model and plot validation line chart and validation scatter plot.
   3) Calculate the corresponding RMSE, RSS and R square value.

8. Building the model of Neural Network, adjusting parameters and plotting graphs.
   Algorithm:
   1) Create a Neural Network model . It has four hidden layers (100, 70, 20, 10), maximum itinerary = 2000, activation function = ReLU, learning rate = 0.01 and solver = adam.
   2) Train the model and plot validation line chart and validation scatter plot.
   3) Calculate the corresponding RMSE, RSS and R square value.

9. Compare these 6 models. Build a graph to compare the RMSE, RSS and R square value.

Link to code in github:
https://github.com/ZaneXiao1/EAEE9305ML-project_zx2407_power-generation-predict

5. Results

5.1 Project Results

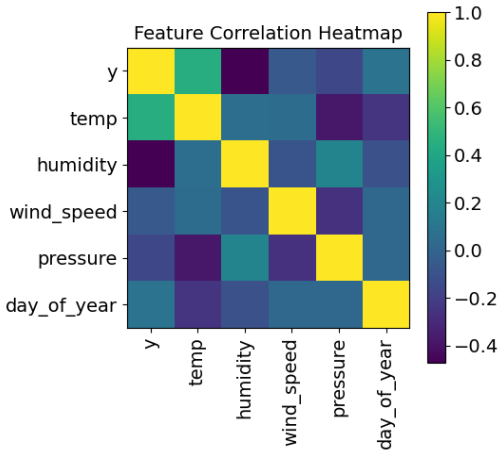


Figure 8. Feature Correlation Heatmap

A feature correlation heatmap (Fig. 8) is a graphical representation used to visualize the correlation between multiple variables at once. From the heatmap above, we can see that the temperature-y color is green, it means that temperature has a relative high positive correlation with y (energy yield). Wind speed and pressure are weakly correlated with capacity. In contrast, the color block between humidity and y is darker, indicating a negative correlation between these two ranges.

The training results of the six models are shown in the following table1. From the magnitude of the R-squared values, it can be seen that ensemble learning models(RF and XGBoost) and NN model perform better than time series ML models (Prophet and LSTM) in terms of model fit. This trend can also be seen clearly from scatter plot (Figure 9).

Table 1. Evaluation index values for the regression algorithms of the six models

Model	RMSE	RSS	Coeff Of Det(R^2)
RF	1803.694855	3.950092e+05	0.693930
NN	2061.462454	4.514603e+05	0.648678
XGBoost	1934.791585	4.237194e+05	0.670265
LSTM_without_weather	4504.647965	1.702757e+06	0.204599
LSTM_with_weather	4942.566908	1.868290e+06	0.127274
Prophet	3049.731413	1.168047e+06	0.473485

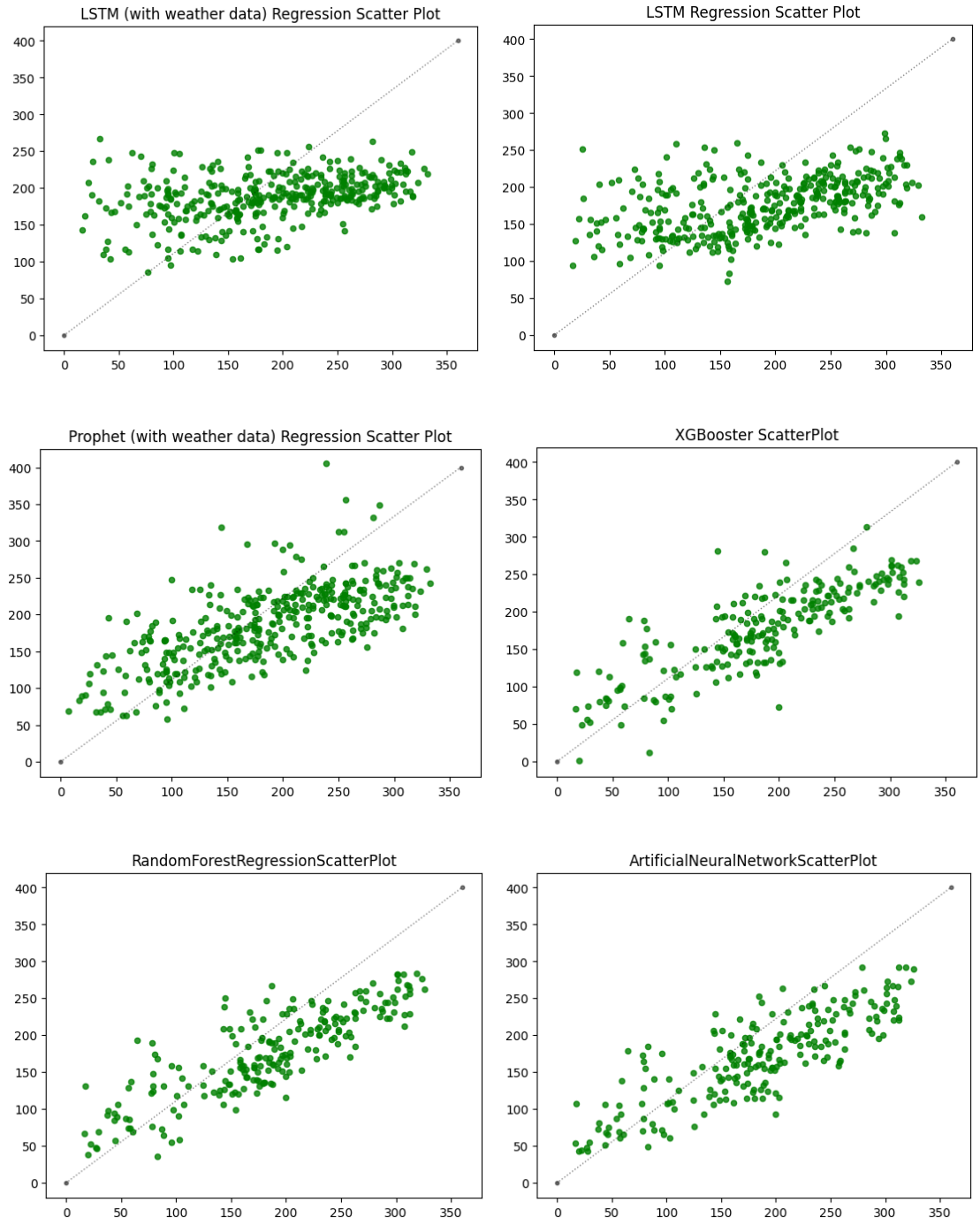


Figure 9. Scatter plot of the six models (The horizontal axis is the actual value and the vertical axis is the predicted value)

We try to adjust the parameters for the best-performing random forest model to optimize its performance further. We reduce the learning rate while increasing the number of estimators or increase the learning rate and decrease the number of estimators. The specific learning rate is varied in tenfold values (0.1, 0.01,

0.001) and the estimator is adjusted in steps of 50 (600, 650, 700, 750, 800). But the performance of my model fit is worse than the initial parameters.

We also try to find out why ensemble learning models (RF and XGBoost) perform better in predicting energy yields compared to time series ML models:

(1) Feature Interactions: Because the relationship between our weather, time and energy yield data might not be linear or mightly depend on specific combinations of weather conditions and times. And RF and XGBoost are particularly good at capturing complex interactions between features.
(2) Dealing with Noise and Outliers: Real-world data like weather and time series data usually have much noise and outliers. And ensemble models like RF and XGBoost are typically more robust to these kinds of noise.
(3) Hyperparameter Tuning: Ensemble learning models like RF and XGBoost are inherently more robust due to their nature, as they combine multiple weak learners to produce a strong learner. This results in better generalization and predictive power, reducing the chances of overfitting.

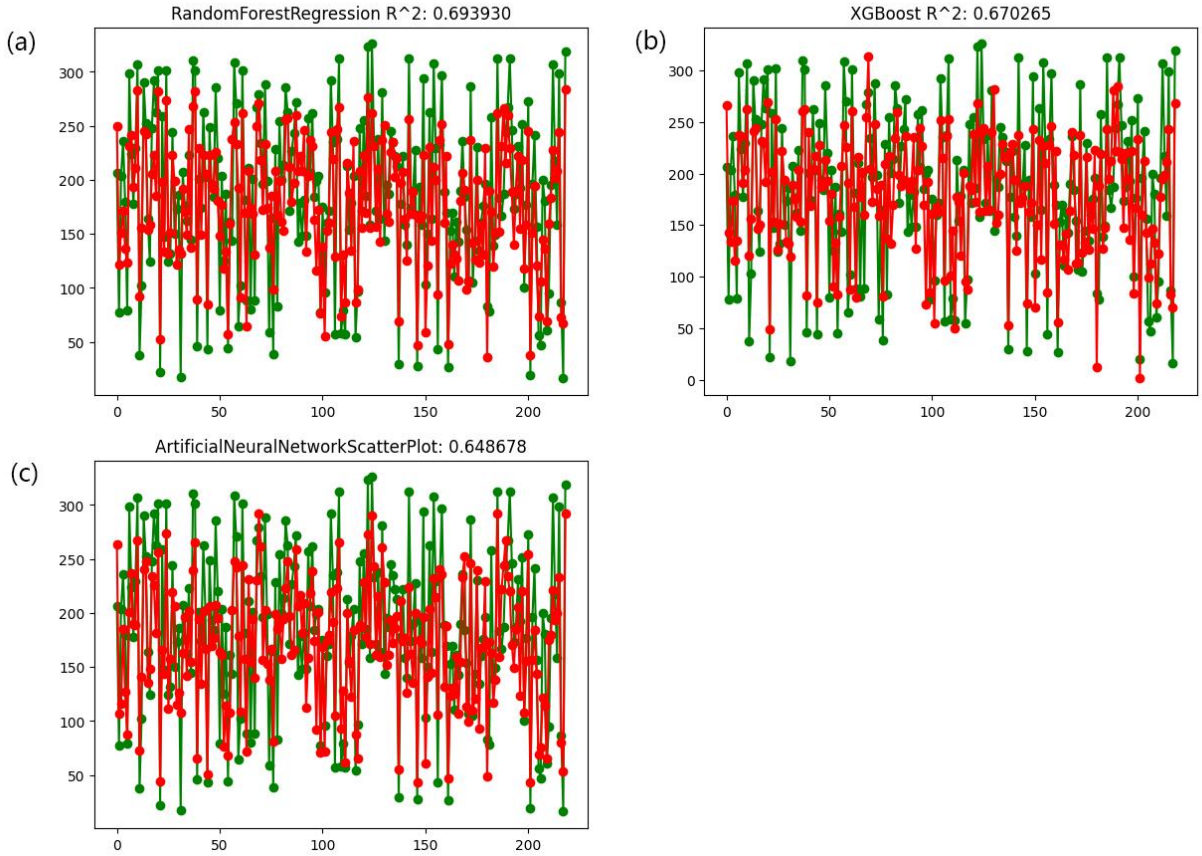


Figure 10. Fitting effect of the three models at the last training (red is the predicted value, green is the actual value)

However, the overall poor performance of the time series model was unexpected, especially the LSTM fit was very poor. So in the next section we will analyze the time series model in detail.

First, for the two sets of LSTM models built, the LSTM model that adds weather data as input has a lower fitting effect than the LSTM model that uses only time as input for prediction instead. We tried to

optimize the models by changing the number of invisible layer cells, the lookback parameter, the learning rate and the activation function. However, the improvement of the fitting results was minimal, and the best r-square value obtained by calibration is 0.2. Thus, we speculate on the following main reasons**[10]**:

(1) Relatively small data set: LSTM models are a type of recurrent neural network and usually require large amounts of data to learn effectively. With a training set size of only about 750 (The dataset provides only three years of solar production data), the model might be underfitting, meaning that it's not be able to learn enough from the data to make accurate predictions.

(2) Feature Selection: Adding more features doesn't always improve the model. If the weather data is noisy or irrelevant, it could actually degrade the performance. Moreover, in the LSTM model, the original look-back process using time series already contains all the factors that affect the output before. Adding the weather with low correlation as the output may instead increase the noise and affect the fitting results. This might be why the model with only time as input performs better than the one with time and weather data.

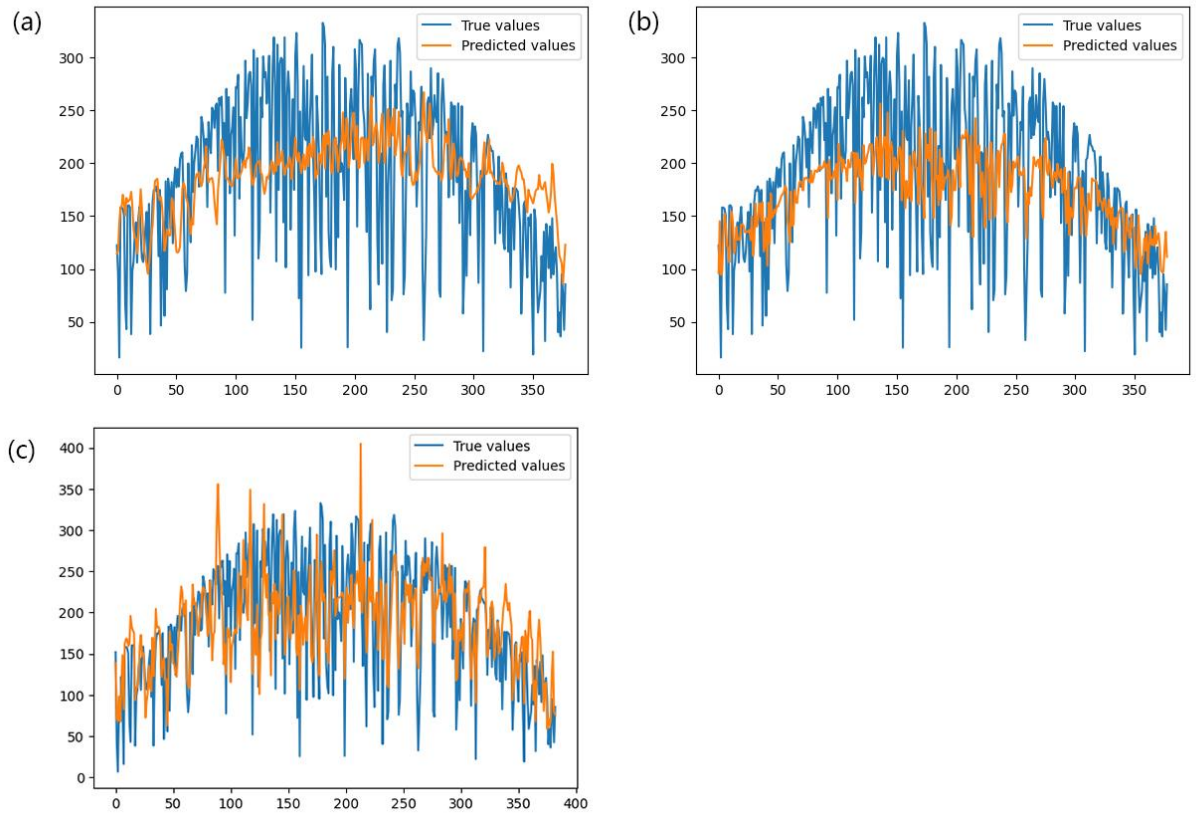


Figure 11. Fitting effect of the three models at the last training (orange is the predicted value, blue is the true value) (a) LSTM(with weather data); (b) LSTM(without weather data); (c) Prophet

The reasons for the differences in the performance of the two different time series models are analyzed next. For the Prophet model, it has an r-square value of 0.47. and the overall trend of its predicted values matches the true values as can be seen from its fitted effect plot. In addition, the training time of Prophet model is shorter than that of LSTM model. We also analyzed several main reasons for these results:

(1) Model Complexity and Data Size: As mentioned above, LSTM is a deep learning model that typically requires large amount of data to perform well. But on the other hand, Prophet is simpler model that can work well even with smaller datasets.
(2) Seasonality feature: Prophet is developed by Facebook, it is specifically designed for time series forecasing with a focus on capturing trend and seasonality. We know that solar energy production varies very significantly from season to season, with the basic rule being less in winter and more in summer. And Prophet can be very effective in modeling these components.
(3) Handling of Regressors: In Prophet, we add weather data as extra regressors. Prophet then fits these regressors in a linear manner, which could be sufficient if the relationship between weather and energy prediction is linear or near-linear. But for LSTM, on the other hand, captures complex, non-linear relationships which might not be necessary or even detrimental if the relationships are actually linear.
(4) Regarding the train time difference: (a) Computational Complexity: LSTM models are more computationally intensive than Prophet models. They involve more complex backpropagation steps and parameter updates, which can take significantly more time; (b) Model Simplicity: Prophet is a simpler model that uses a decomposable time series model combining trend and seasonality. This is much less computationally demanding than the LSTM architecture.

5.2 Comparison with last semester's project

Last semester's project also used weather and climate data to make predictions of solar power production (Using XGBoost, RF and NN models). However, there are major differences in the dataset and the range of predictions used in the two projects. The specific differences are reflected in the fact that the previous project's forecast location was solar power production for the entire state of California, i.e. it was mainly energy production forecasts for large solar power plants. The corresponding weather data is also the average temperature for the entire state of California. Time is measured in months, and six consecutive years of data are used as the dataset.

The following Table 2. shows the model fitting results of the last project.

	Model	RMSE	RSS	Coeff Of Det(R^2)
0	RF	2.152398e+08	5.531663e+10	0.680998
1	NN	2.198799e+08	5.650913e+10	0.674118
2	XGBoost	1.682934e+08	4.325141e+10	0.750574

Table 2. Evaluation index values for the regression algorithms of the three models (Predict Monthly Solar Production in California)

The prediction performance of the models in both projects is similar in terms of the fitting effect. Both were able to train machine learning models with high predictive power by using smaller data sets as inputs. In contrast, the model used in the previous study had a better fit. We speculate that there may be some reasons for this: (1) Differences in dataset size: Although the previous project predicted a larger area of the entire state of California and used fuzzier averages of weather data. However, it used a more extensive dataset of six years, which is twice the size of the current dataset. A larger dataset can give a richer training effect to the model. (2) Differences in dataset quality: The data used for California are from the US Energy Information Administration (EIA) and the National Oceanic and atmospheric administration, measured by professional weather stations and energy monitoring equipment. The data used in this study are open-source data from solar panel suppliers, which may be less accurate.

Overall, the results of this study and last semester's project together demonstrate that the ensemble learning model performs relatively well for both large-scale solar development plants and small-scale home solar panel capacity forecasting.

5.3 Discussion

The project was generally completed as expected, but I encountered some challenges and gained some experience throughout the process.

In this research project, the time series model performed much less well than the ensemble model.
By changing the model parameters through continuous parameter tuning, the time series model still performed poorly in terms of fitting. This is where the selection and pre-processing of the data set should be considered. During the first training, I input the data into the LSTM model without processing it with the normalization function, resulting in a very poor fit. Therefore, the normalization operation is very critical in the training of LSTM models to eliminate the bad effects caused by odd sample data. In addition, the overall data set of the time series model is also more demanding.

In addition to this the time series model has certain requirements for the whole data set. When constructing a time series model there are several key requirements the dataset should ideally meet. For example, the following requirements**[11]**- Enough data: The more data points you have, the more reliable your time series model will be. Having data over a longer period allows your model to better learn and understand trends and seasonality in the data; Order: Observations should be arranged in chronological order. The order of the data is important in time series analysis because it uses past data to predict future data; Consistent intervals: The interval between data points should be consistent. For example, if it is a daily time series, there should be an observation point for each day, and no day should be skipped. Gaps in the data can lead to inaccurate predictions.

The above requirements where the order can be done by data pre-processing operations, but the data set size cannot be changed by data processing. So when considering using a time series model, the data set size is the point that needs to be prioritized.

6. Future Work

We have verified in this research project that the ensemble model has good performance properties for both small and large scale predictions of solar energy. While the time series model performs poorly. For future research, it is recommended to use datasets measured by professional institutions and to increase the dataset size to meet the requirements of the time series model. In addition, the accuracy of the model will be higher if more relevant data on solar energy production is available, such as solar radiation and solar panel power records. Model training on the above conditions is expected to produce better prediction models.

7. Conclusion

In this project, we propose machine learning (ML) strategies that use time series models (LSTM, Prophet), ensemble learning models (XGBooster, Random Forest), and neural network models to predict daily energy production. Model training is performed using real open source data from Sydney, Australia. In our results, the ensemble model shows excellent prediction performance, which is in line with the results of the referenced literature, and also validates the ensemble model as an efficient model, both in the area of predicting solar energy production for homes and predicting electrical energy production for large-scale solar power plants. In contrast, the time series model performs poorly, and we also analyze the possible reasons and means of improvement. Through this research project, we learned to use code to build multiple machine learning models and to infer the causes backwards from the model results. Experience was also gained on the different requirements of various models for data sets. We believe that

machine learning techniques still have a wide scope for development in the field of sustainable energy prediction.

8. Acknowledgement

During this project, I want to express my acknowledgement to the PHD Weiwei Zhan and Professor Pierre Gentine. It is your help that this project could stay on track. I also want to thank the referenced research. Their paper has been a huge inspiration on my project. In addition, online resources also give me some different angle solutions for the problems I am facing.

9. References

[0] Project github link:
https://github.com/ZaneXiao1/EAEE9305ML-project_zx2407_power-generation-predict

[0] *Solar Home Electricity Data*. Ausgrid. (n.d.). https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data

[1] Iea. (n.d.). *Renewable electricity – analysis*. IEA. https://www.iea.org/reports/renewable-electricity

[2] Barrera, J. M., Reina, A., Maté, A., & Trujillo, J. C. (2020, August 25). *Solar energy prediction model based on artificial neural networks and open data*. MDPI. https://www.mdpi.com/2071-1050/12/17/6915

[3] Forecasting process in FbProphet and Neural Prophet. (n.d.). https://www.researchgate.net/figure/Forecasting-process-in-FbProphet-and-neural-prophet_fig3_359290223

[4]Rahuljha. (2020, June 29). *LSTM gradients*. Medium. https://towardsdatascience.com/lstm-gradients-b3996e6a0296

[5] *What is XGBoost?* NVIDIA Data Science Glossary. (n.d.). Retrieved December 23, 2022, from https://www.nvidia.com/en-us/glossary/data-science/xgboost/

[6] *What is Random Forest?* IBM. (n.d.). Retrieved December 23, 2022, from https://www.ibm.com/topics/random-forest

[7] *RSS, MSE, RMSE, RSE, TSS, R2 and adjusted R2*. RSS, MSE, RMSE, RSE, TSS, R2 and Adjusted R2 - Yao's blog. (n.d.). Retrieved December 22, 2022, from https://blog.listcomp.com/machine-learning/2014/09/29/residual-error-term-rss-mse-rse-rmse-tss-r2-and-adjusted-r2

[8] *Solar Home Electricity Data*. Ausgrid. (n.d.). https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data

[9] *Local weather forecast, news and Conditions*. Weather Underground. (n.d.). https://www.wunderground.com/

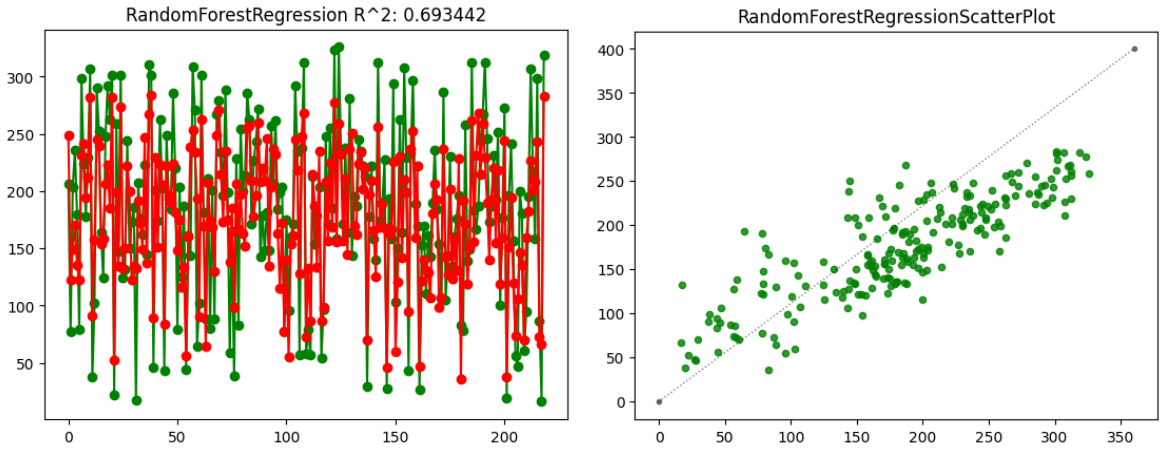
[10] Montantes, J. (2020, February 4). *3 reasons to use random forest over a neural network–comparing machine learning versus deep...* Medium. https://towardsdatascience.com/3-reasons-to-use-random-forest-over-a-neural-network-comparing-machi

ne-learning-versus-deep-f9d65a154d89#:~:text=Random%20Forest%20is%20less%20computationally,tree%20but%20with%20better%20performance.

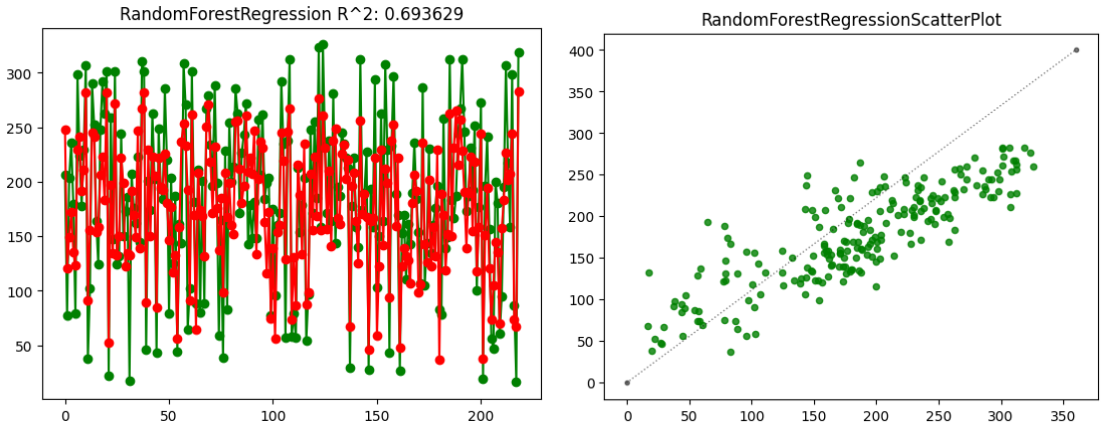
[11] Chowdhury, K. (2021, May 25). *10 hyperparameters to keep an eye on for your LSTM model-and other tips*. Medium. https://medium.com/geekculture/10-hyperparameters-to-keep-an-eye-on-for-your-lstm-model-and-other-tips-f0ff5b63fcd4#:~:text=Widely%20accepted%2C%20a%20good%20default,your%20LSTM%20or%20RNN%20model.

10. Appendix

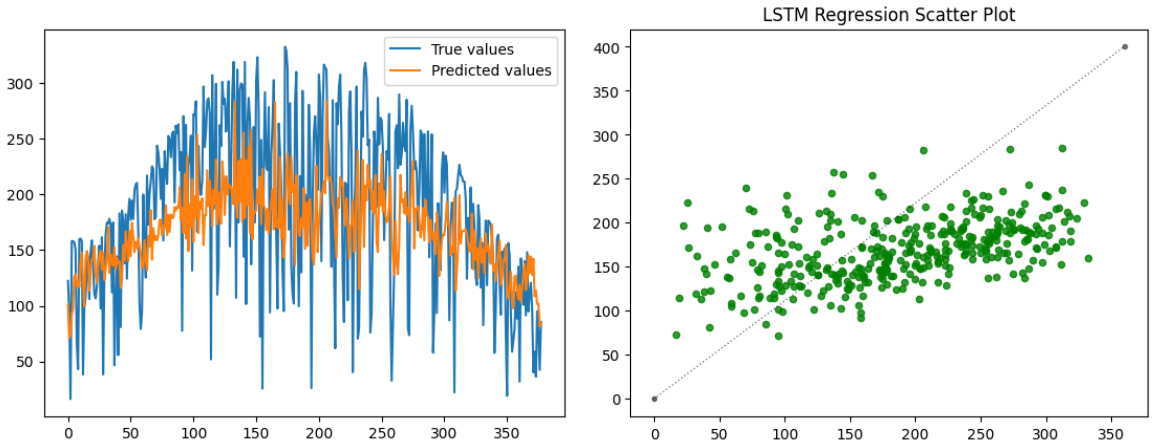
The following is a graph of the change in the fitting results after parameter adjustment:



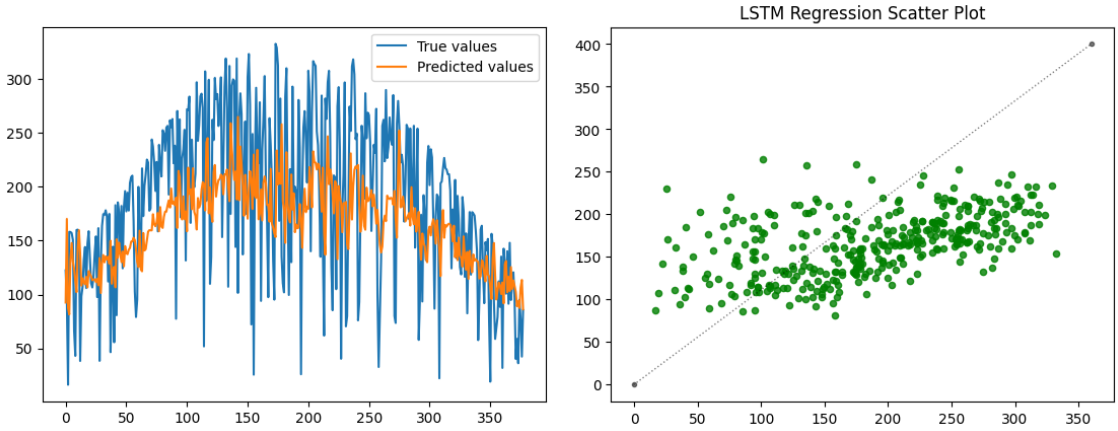
A Figure 1. Scatter plot and fitting effect plot of RandomForest Model (Increase n_estimator to 1000)



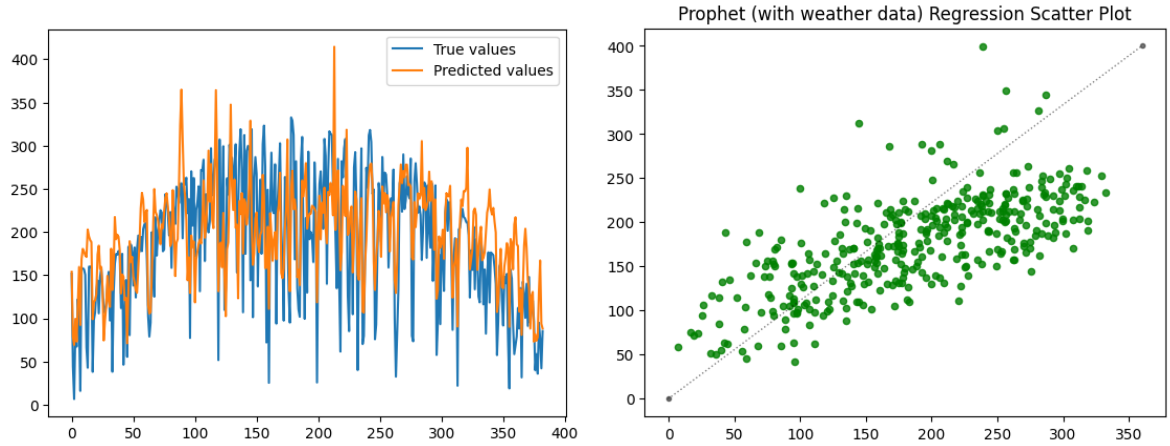
A Figure 2. Scatter plot and fitting effect plot of RandomForest Model (Decrease n_estimator to 500)



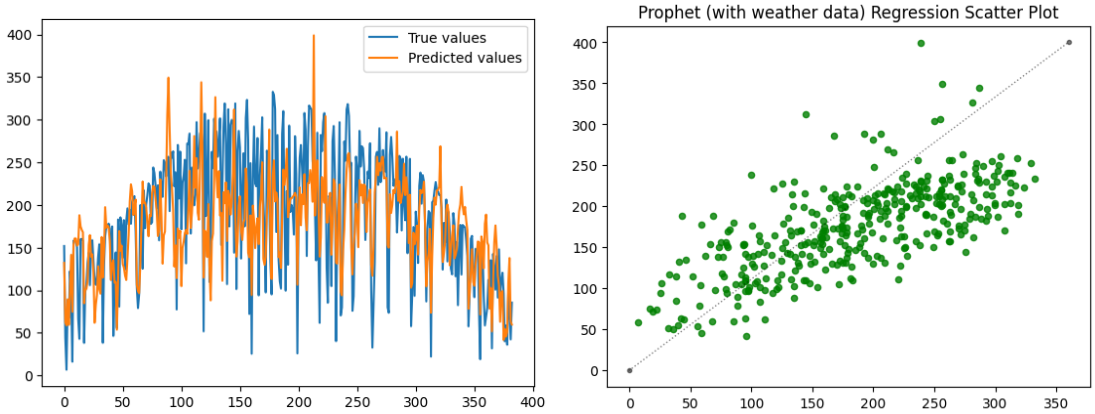
A Figure 3. Scatter plot and fitting effect plot of LSTM Model (Decrease learning_rate to 0.0001)



A Figure 4. Scatter plot and fitting effect plot of LSTM Model (Increase learning_rate to 0.01)



A Figure 5. Scatter plot and fitting effect plot of Prophet Model (Decrease Change_point to 0.001)



A Figure 6. Scatter plot and fitting effect plot of Prophet Model (Increase Change_point to 0.01)