# Using Three Machine Learning Models to Predict Daily Solar Production in California

EAEE4000.zx2407.report
Zihao Xiao zx2407
*Columbia University*

**ABSTRACT**
*In this project, we build three kinds of machine learning models (XGBooster, Random forests, and Neural Networks) to predict daily solar yields in California. We adjusted the parameters to improve the performance of the models. Then we compared the results of these three models. Among our results, XGBooster shows the best fit. And the neural network performs relatively poorly. We analyzed the possible reasons for their performance. The conclusions we draw are generally consistent with the original paper and achieve our expected goal.*

## 1. Introduction

Due to the further increase in demand for environmentally friendly clean energy, forecasting its production has become increasingly important. Solar energy is a renewable energy source that is derived from the sun's rays. It is an important source of electricity for homes, businesses, and communities around the world. And solar energy is growing strongly in terms of its production, with California, for example, which is the most abundant producer of solar energy in the entire United States, producing a total of 33,670 gigawatt-hours (GWh)[1] in 2021 alone. So the need to have predictive models for solar energy production is increasing.

Due to the rapid development of artificial intelligence, machine learning-based prediction models have been reported for various energy-related production forecasts, for example, for municipal electricity production. However, some of the models apply limited data sets to predict energy production over large areas, such as predicting the energy production of an entire country. However, we know that solar energy production is affected by a combination of conditions such as topography and climate latitude, and there are differences between regions. So modeling for a limited range may be more relevant and accurate.

In this study we propose an ML strategy to predict solar yield by month, temperature and rainfall. The study area we selected is California, which is typical and has a large solar energy production capacity. We build three different machine learning models (Random Forest, XGBooster and Neural Network). We compared the prediction performance of these three models and analyzed the reasons. This information should be valuable for the future application of ML in the energy field.

## 2. Methodology

In this project, we used a machine learning regression algorithm. The inputs to the model are month, temperature and rainfall, and the output is the solar yield for the day. The main methods designed in this paper are three machine learning models, namely XGBooster, Random Forest and Neural Network. So in the following section, we will introduce these three models. The specific code implementation is described in the next section.

(1) XGBooster[2] (eXtreme Gradient Boosting) is an ensemble learning method for classification and regression problems in machine learning. It is an extension of the gradient boosting algorithm and is an improvement over the traditional gradient boosting method. XGBooster works by training a series of decision tree models in a sequential manner, with each tree trying to correct the mistakes of the previous tree. The final

prediction is made by combining the predictions of all the trees in the ensemble.The flowchart of the XGBooster is listed below (Figure 1).

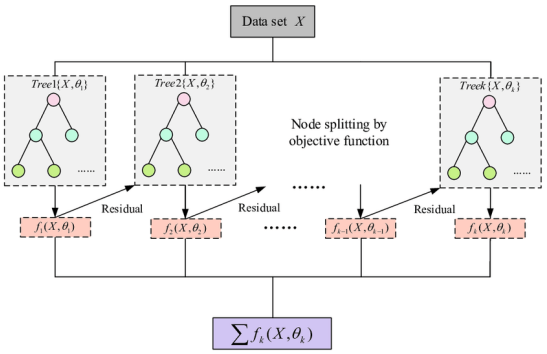


Figure 1. Flowchart of the XGBooster[3]

The principle of XGBooster can be described as follows: Initialize the weights of the training data points: The weights of the training data points are initialized to a uniform value; Train the first decision tree: A decision tree model is trained on the training data with the initialized weights; Update the weights of the training data points: The weights of the training data points are updated based on the prediction error of the first decision tree. The data points that are predicted correctly by the first decision tree have their weights reduced, while the data points that are predicted incorrectly have their weights increased; Train the second decision tree: The second decision tree is trained on the updated weights of the training data points; Update the weights of the training data points: The weights of the training data points are again updated based on the prediction error of the second decision tree; Repeat the process: Steps 4 and 5 are repeated until the desired number of decision trees is reached; Make the final prediction: The final prediction is made by combining the predictions of all the decision trees in the ensemble using a weighted average, where the weight of each tree is based on its prediction error.

(2) Random Forest[4] is another popular ensemble learning method for classification and regression problems in machine learning. It works by training a large number of decision tree models on randomly selected subsets of the training data, and then combining their predictions to make the final prediction. The decision tree models are trained in parallel, and the final prediction is made by taking a majority vote of the predictions of all the decision trees. The flowchart of the Random Forest is shown below (Figure 2).

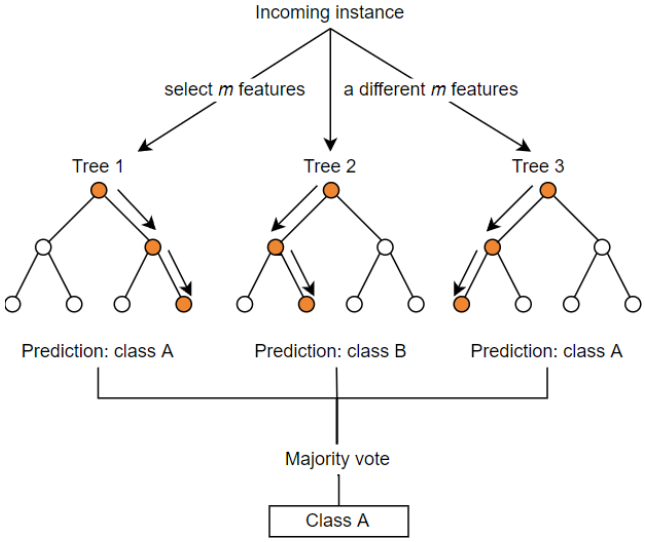


Figure 2. Flowchart of the Random Forest[5].

The principle of Random Forest can be described as follows: Randomly select K subsets of the training data: K subsets of the training data are randomly selected, with replacement, from the original training data; Train a decision tree on each subset: A decision tree model is trained on each of the K subsets of the training data; Make the final prediction: The final prediction is made by taking a majority vote of the predictions of all the K decision tree models.

(3) Neural Network[6] is a machine learning model inspired by the structure and function of the human brain. It consists of layers of interconnected neurons, where each neuron receives input from the previous layer, processes the input using an activation function, and then passes the output to the next layer. Neural networks can be trained to perform a wide variety of tasks, including classification and regression. The flowchart of the Neural Network is shown below (Figure 2).

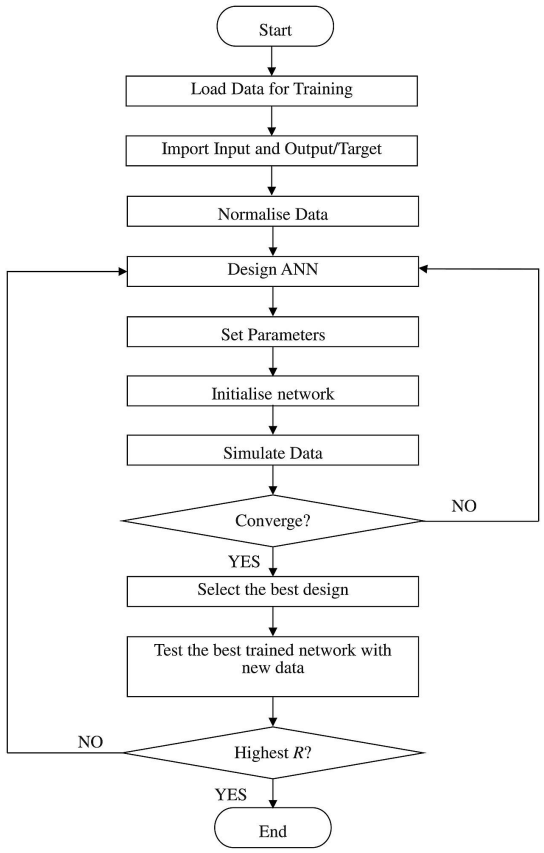


Figure 3. Flowchart of the Neural Network[7].

The principle of Neural Network can be described as follows: Initialize the weights and biases of the neurons: The weights and biases of the neurons in the neural network are initialized to random values; Forward propagation: The input data is passed through the neural network, starting from the input layer and going through the hidden layers, until it reaches the output layer. At each layer, the input is processed by the neurons using the weights and biases, and the output is passed to the next layer; Calculate the prediction error: The prediction error is calculated based on the difference between the predicted output and the actual output; Backward propagation: The prediction error is propagated.

In addition to machine learning models, in the analysis section we use some evaluation metrics of regression algorithms[8]:

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-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 three machine learning model frameworks in the Anaconda environment.
(2) Training models and tuning parameters to improve model accuracy.
(3) Comparing the performance of the three models, trying to analyze the reasons and summarize the project gains.

Technical Challenges:
(1) There is no matching energy production data and weather data on the network. So I need to find the corresponding data separately and

match the data manually. This process will take a lot of time.
(2) Some of the data for factors that have a large impact on solar production were not found, for example, daily solar radiation for the California region was not found on the web. The lack of such data may affect the accuracy of the final model fit.
(3) It is necessary to understand the principles of the three machine learning models and how the code is implemented, and how to adjust the parameters to improve the fit during the training of the model.

## 4. Implementation

The flow chart below shows the progresses of this project.(Figure 4)

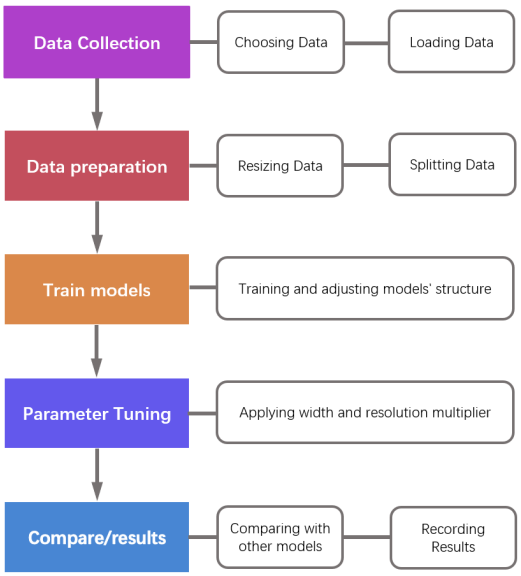


Figure 4. Flowchart of the progresses of this project.

### 4.1 Data

The data we use is from Natinal Weather Service of the National oceanic and atmospheric administration and US Energy Information Administration (EIA). From these website we can find the daily temperature, precipitation and solar energy produciton data from 1990 to now. Because the memory limitation of our calculation devices and to shorten the training time of the model, we have decided to pick only the data from the recent four years (from 2018 to 2021). The total number of data we use is 1272. The table below shows the format of input data we upload to the environment.

Since statistical data are inevitably subject to measurement error, the data are to be imported. We will plot the point distribution of the data and remove the scatter that deviates significantly from the distribution.(Figure 4)

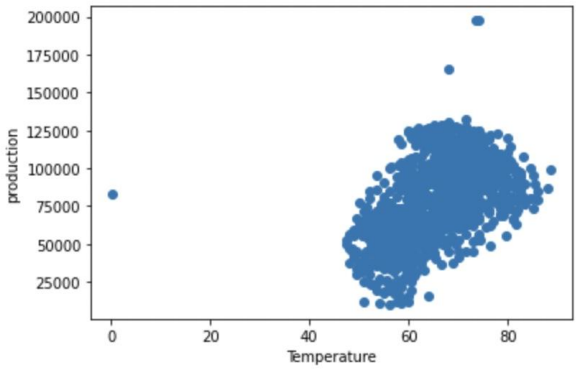


Figure 5. Distribution of solar energy production in relation to temperature.

### 4.2 Implementation of the three models

Our implementation of XGBooster contains four manually set parameters: learning rate, Minimum sum of instance weight needed in a child; Maximum depth of a tree; The number of estimators in XGBooster.

For Random Forest we set: The number of estimators in Random Forest and random state.

For Neural Network we set:Structures of Hidden Layers, Learning rate, Maximum itineration, activation and solver.

### 4.3 Software Design

We have implemented a total of 3 models:

1. Building the model of XGBooster, adjusting parameters and plotting graphs.
   Algorithm:
   1) Load training and testing data set.
   2) Scatter plots of yield and month, rainfall and temperature related distributions.

3) Remove deviations where the temperature is greater than 60 and the solar output is greater than 15,000; remove deviations where the temperature is less than 10.
4) The yield column is removed from the data and the rest form the input value; the yield column is retained from the data and forms the output value.
5) Randomly separated test and training groups. Testsize = 0.2, randomstate = 50.
6) Display the distribution of training set and testing set (Figure 6).
7) Create a XGBooster model with estimators = 500, learning rate = 0.03, minimum child weight = 5 and maximum depth = 2.
8) Train the model and plot validation line chart and validation scatter plot.
9) Calculate the corresponding RMSE, RSS and R square value.
10) Decrease learning rate = 0.01, increase estimators = 700.
11) Train the model and plot validation line chart and validation scatter plot.
12) Calculate the corresponding RMSE, RSS and R square value.
13) Increase learning rate = 0.1, and decrease estimators = 200.
14) Train the model and plot validation line chart and validation scatter plot.
15) Calculate the corresponding RMSE, RSS and R square value.

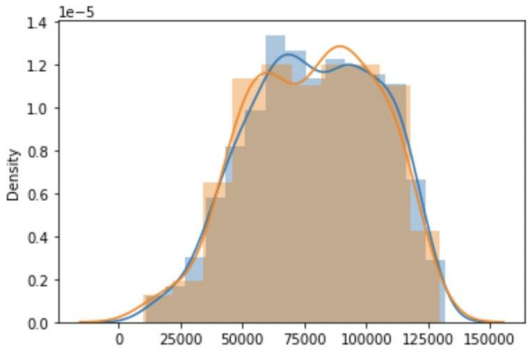


Figure 6. Distribution of the training and test sets (orange is the test set, blue is the training set)

2. Building the model of RandomForest, adjusting parameters and plotting graphs.
   Algorithm:
   1) Divide the testing set and training set to the shapes of (257, 3) and (1025,3).
   2) Create a XGBooster model with estimators = 800, learning rate = 0.03, random_state = 0.
   3) Train the model and plot validation line chart and validation scatter plot.
   4) Calculate the corresponding RMSE, RSS and R square value.

3. Building the model of Neural Network, adjusting parameters and plotting graphs.
   Algorithm:
   1) Divide the testing set and training set to the shapes of (257, 3) and (1025,3).
   2) 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.
   3) Train the model and plot validation line chart and validation scatter plot.
   4) Calculate the corresponding RMSE, RSS and R square value.

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

Link to code in github:

1. https://github.com/ZaneXiao1/EAEE4000ML-project_zx2407_power-generation-predict

## 5. Results

### 5.1 Project Results

The training results of the three models are shown in the following table. From the magnitude of the R-squared values, it can be seen that XGBooster performs the best in terms of model fit, reaching 0.75. It is followed by the random forest model with an R-squared value of 0.67, and the last in line is the neural network with 0.68. This trend can also be seen in Figure 7 and Figure 8.

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

| | 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 |

For the XGBooster model, which has the best performance, we try to change its parameters to optimize it. In the first group, we reduce the learning rate to 0.01 and correspondingly increase the number of estimators to 700; in the second group, we reduce the number of estimators to 200 and increase the learning rate to 0.1. However, from the observed results, the fit is poor compared to the initial settings.

We also checked some literature on the web for the reasons why XGB performs better. We have summarized a few possibilities:

(1) XGBooster is a boosting algorithm, which means that it combines the predictions of many weak learners (i.e., decision trees) to produce a strong overall model. Boosting algorithms can often achieve better performance than a single decision tree, especially on imbalanced or hard-to-classify datasets.

(2) XGBooster can handle missing values and categorical variables natively, which can be helpful if your dataset has these types of data.

As for the reasons why neural networks perform poorly, we likewise summarize a few possible reasons:

(1) Insufficient data: Neural networks require a large amount of data to train, and they may not perform well if the data is not sufficient[9]. And we are using a small dataset, so it is possible that XGBooster or Random Forest might be more suitable models.

(2) Overfitting: Neural networks are prone to overfitting if the model is not properly regularized, especially when the data is not sufficient. This may also lead to poor performance on our test set.

(3) Poor model architecture: The performance of a neural network depends heavily on the architecture of the model, including the number of layers, the number of neurons in each layer, and the choice of the activation function. It is possible that our neural network's architecture is not suitable for the task, and it lead to poor performance.

(4) Hyperparameter tuning: Neural networks have many hyperparameters that need to be properly tuned in order to achieve good performance. If the hyperparameters are not set optimally, it can lead to poor performance.

We have also made many attempts for the possibilities above to improve the Nerual Network models. These include adjusting the learning rate and the maximum number of iterations in the parameters; modifying the number of hidden layers and the number of nerves in each layer; and adding early stopping to prevent overfitting. However, the improvement is small and the fitting results are still not as good as the other two models. So

finally, we guessed that the poor performance of the neural network model was more likely due to the insufficient amount of training data.

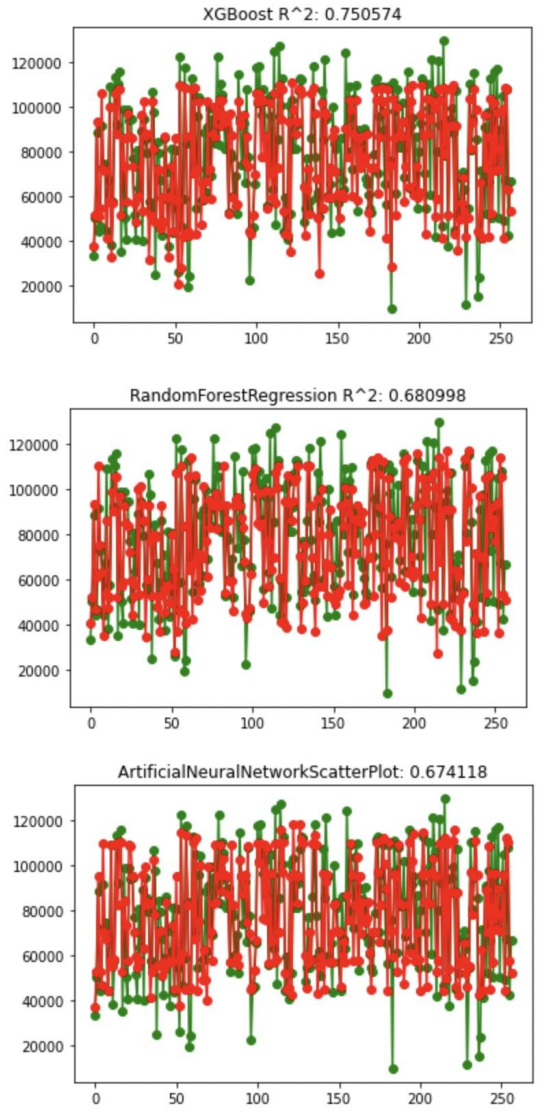


Figure 7. Fitting effect of the three models at the last training (red is the predicted value, green is the actual value)

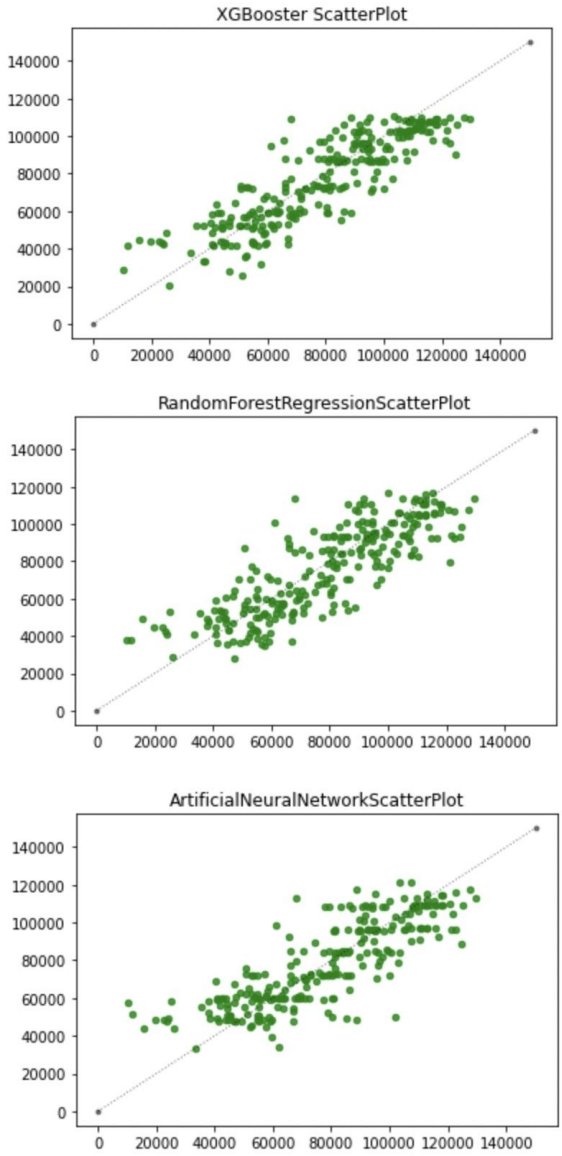


Figure 8. Scatter plot of the three models (The horizontal axis is the actual value and the vertical axis is the predicted value)

## 5.2 Discussion

The project was generally accomplished as expected, but I encountered a number of challenges throughout the process.

One of them is that when I first built the model to train the dataset, all three models showed very poor fitting performance. This was evident in the significant deviation from the high solar energy production region, as shown in the Figure 9. I initially tried to adjust the learning rate, number

of iterations, and other parameters, but there was no significant improvement. So I had to suspect that there might be a problem with my input dataset, and I proceeded to output the distributions of the training and test sets to the same graph (Figure 10). I found that the test set showed significant peaks in the high-yield region, which was very different from the distribution of the training set.

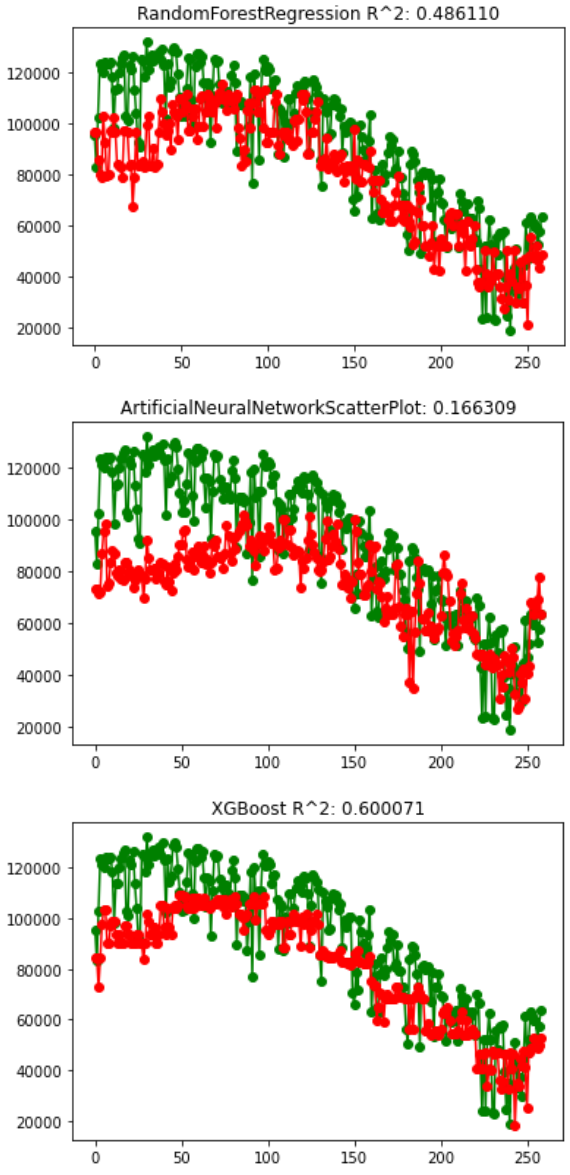


Figure 9. Fitting effect of the three models at the first failed training (red is the predicted value, green is the actual value)

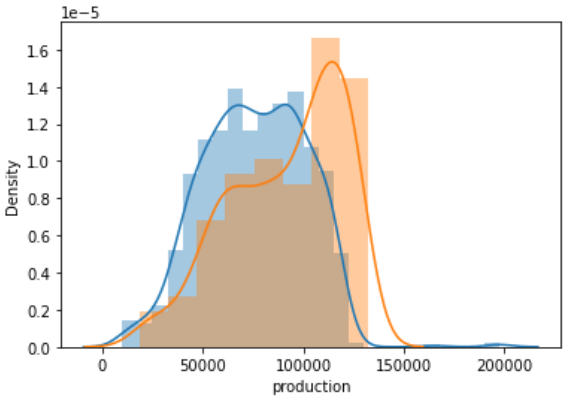


Figure 10. Distribution of the training and test sets at the first training (orange is the test set, blue is the training set)

So I started to process the data again, this time I first combined the previously divided training and test sets. Then I regrouped them, this time adding random parameters so that all the data in the new test and training sets were randomly selected and not arranged linearly along time. The experimental results proved that the model fit was significantly improved after the second attempt.

## 6. Future Work

The expepriments we did in this project is based on a relatively small dataset with 1272 data groups. A possible improvement would be to train these models (especially Neural Network model) on a larger dataset and compare the results again. Furthermore, more factors that affect the amount of solar energy produced can be added as inputs, such as the amount of solar radiation and the temperature effectiveness of the solar panels. Besides, more experiments on parameter tuning could be conducted, for example, adjust the learning rate of the optimizer and the dropout rate, try to add regularization to some layers, in order to accelerate the training process, prevent overfitting, and producce a better result.

## 7. Conclusion

In this project, we built machine learning models to predict solar energy production based on the principles of XGBooster, random forests

and neural networks. We adjust parameters to improve the performance of the models. We then compare the results of these three models. In our results, XGBooster shows the best fit. And neural networks perform relatively poorly. The conclusions we reached were generally consistent with the original paper and met our expected goals. The overall trend of the results with other literature on projected energy production is consistent and meets our expectations. Based on our results we also analyze the possible causes. Through this project, we learned how to build multiple machine learning models using code and optimize the model structure by analyzing the results backwards. We believe that machine learning will have a broader research potential in the field of energy, especially in the field of clean energy represented by solar energy.

## 8. Acknowledgement

During this project, I want to express my acknowledgement to the Teaching Assistant 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/EAEE4000ML-project_zx2407_power-generation-predict

[0] Data source: https://www.cnrfc.noaa.gov/?lat=36.361&lng=-114.693

[1] McDonnell, S. (2022, November 14). *Meet two of the biggest solar farms in the U.S.* LeafScore. Retrieved December 23, 2022, from https://www.leafscore.com/blog/meet-two-of-the-biggest-solar-farms-in-the-u-s/

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

[3] Guo, R., Zhao, Z., Wang, T., Liu, G., Zhao, J., & Gao, D. (2020). Degradation state recognition of piston pump based on ICEEMDAN and XGBoost. *Applied Sciences*, *10*(18), 6593. https://doi.org/10.3390/app10186593

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

[5] *Random forests*. DeepAI. (2020, September 10). Retrieved December 22, 2022, from https://deepai.org/machine-learning-glossary-and-terms/random-forest

[6] Abdel-Nasser Sharkawy. (2020). Principle of neural network and its main types: Review. *Journal of Advances in Applied & Computational Mathematics*, *7*, 8–19. https://doi.org/10.15377/2409-5761.2020.07.2

[7] Public Library of Science. (n.d.). *Transformer incipient fault prediction using combined artificial neural network and various particle swarm optimisation techniques*. PLOS ONE. Retrieved December 22, 2022, from https://journals.plos.org/plosone/article/figure?id=10.1371%2Fjournal.pone.0129363.g001

[8] *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

[9] *4 disadvantages of Neural Networks and deep learning*. Built In. (n.d.). Retrieved December 23, 2022, from https://builtin.com/data-science/disadvantages-neural-networks

## 10. Appendix

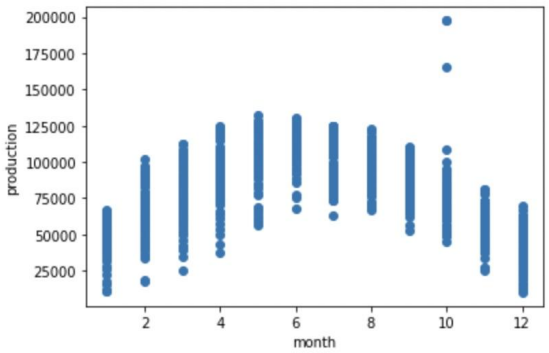


Figure A1.. Distribution of solar energy production in relation to months.

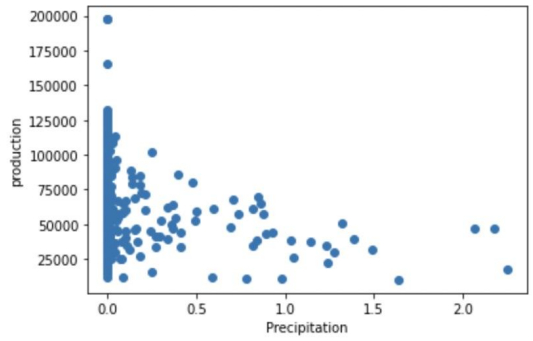


Figure A2. Distribution of solar energy production in relation to Precipitation.

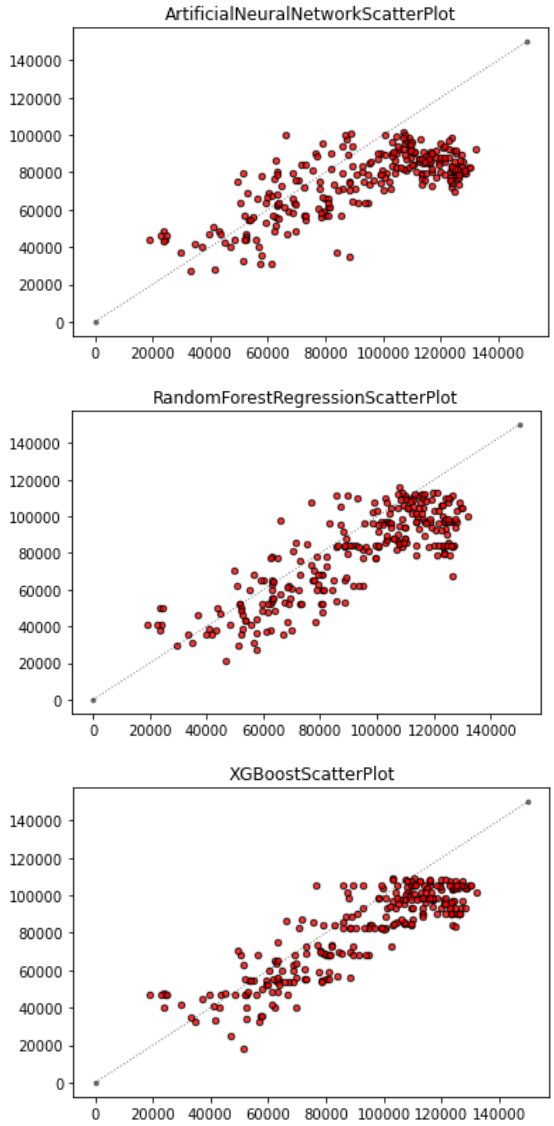


Figure A3. Scatter plot of the three models in the first failed trainning (The horizontal axis is the actual value and the vertical axis is the predicted value)