CS 228 Spring'23 Project Final Report: Solar Power Generation Prediction

Manish Chugani
University Of California, Riverside
Computer Science
862388066
mchug002@ucr.edu

Sumedha Girish Atreysa
University Of California, Riverside
Computer Engineering
862395753
satre002@ucr.edu

Alisha Kulkarni
University Of California, Riverside
Computer Science
862315043
akulk018@ucr.edu

ABSTRACT

In a world with fossil fuels depleting rapidly, the need for technology and science to evolve and ensure that these resources are being maintained efficiently has gone up. In recent years, the shift to using renewable sources of energy for the generation of electricity has drastically transformed the landscape of the electrical power generation market. This work presents a comparison of the statistical data mining techniques and deep learning methods that can be employed for the accurate predictive maintenance of a solar panel grid. The pipeline proposed utilizes input data from sensors placed in a solar power generation plant to make predictions on the power generated by the plant at any time of the day. Malfunctioning inverters can be detected by observing discrepancies in the recorded values and the predicted values of the power generated. This enables efficient maintenance and allows the administrative staff at the power plant to handle these issues with ease.

KEYWORDS

Solar Power Generation, Statistical Data Mining, Deep Learning, Gaussian Process Regression, Support Vector Regression, Neural Networks

1 INTRODUCTION

Solar energy is a source of clean and renewable energy which makes it extremely significant in today's day and age for generating electricity. Despite this, solar power accounts for only 3.6% of the power generated across the globe. This is primarily attributed to the fact that the machinery involved in generating solar power is inefficient in some ways. The efficiency of these power plants can be improved by predicting the generation of solar power. This would subsequently improve the production of solar power, lead to better grid management and quicken the identification of faulty equipments in the system.

There are several challenges to harnessing solar energy because it is dependent on factors such as weather conditions, the conversion rate of solar radiation to electricity, cost of maintenance of equipment, and landscapes, to name a few. Deriving patterns from historic data with efficiency becomes an even important task to ensure reductions in maintenance costs while optimizing the process of harnessing solar energy. Powerful deep learning and data mining techniques can aid in mitigating mishaps and plan maintenance of grids ahead of time. Such techniques can benefit both the consumers and the equipment manufactures of these solar panels. This paper aims to compare a variety of data-driven estimators to arrive at a conclusive methodology with the least possible amount of errors to prevent the wastage of precious resources while maintaining productivity.

The proposed approach uses the dataset titled Solar Power Generation[1]. The dataset consists of two pairs of files containing data from two solar power plants over a 34 day period. The first pair of files contains sensor readings and provides information about the amount of power generated by the plant through measures like ambient temperature, module temperature and irradiation. The second pair of files contains data gathered at the inverter level, where every inverter has several solar panel lines attached to it. It contains data related to the amount of AC and DC power generated at 15 minute intervals along with the daily yield and total yield of the power plant. The daily yield can be defined as the cumulative sum of the generated power on a specific day until a certain point in time. The total yield can be defined as the cumulative sum of the power generated by the inverter on a specific day until a certain point in time. The paper is organized in the following way in the subsequent sections:

(1) Related Work compares the existing methodologies on Solar Power Generation Prediction
(2) The Proposed Methods section compares and contrasts the 2 statistical data mining algorithms and one deep learning algorithm that have been used to predict the solar power generated given the time of day, ambient temperature, module temperature and irradiation from the sensor in the power plant.
(3) The Experimental Evaluation section not only provides an exhaustive comparison of the methodologies but also shows the statistical significance of the results obtained by each method for the task at hand using a two-tailed Student-T Test.
(4) The Discussion & Conclusion section provides an overview of the proposed approaches and also tackles the future work that can be performed for Solar Power Generation.

CS228 S23, 15.00

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1] https://www.kaggle.com/datasets/anikannal/solar-power-generation-data

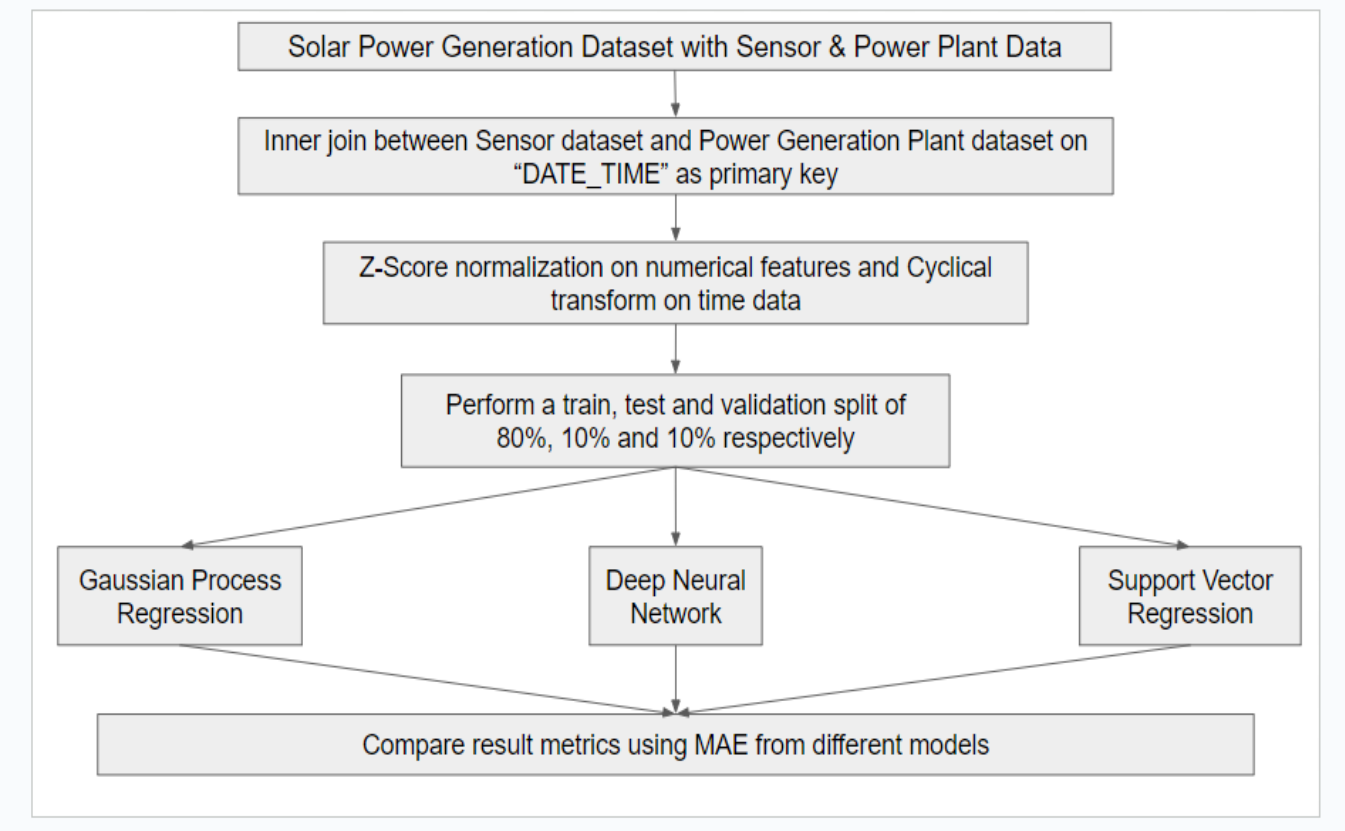

Figure 1: Project Pipeline Overview

2 RELATED WORK

The related work for the prediction of solar power generated is vast and consists of literature and problem statements that utilize deep learning based methods such as, Long Short-Term Memory (LSTM)[1], Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU) and Time Sequential Predictive Models[3] among many others.

The work done by Mariam AlKandari et.al[1] predicts the power generated by a solar panel a day ahead, given the weather forecast and the condition of the solar panel on the current day. While this approach is slightly different from the one presented in this paper, the techniques utilized can be useful for the preventative maintenance of the solar panel. The paper reports the mean absolute error and so it is used to compare against the proposed methodology. The work presented makes use of a GBRT model which is further expanded to include regression, while being developed primarily for classification. GBRT is a machine learning model that combines the output of a number of smaller regression trees of a defined size. This technique helps improve the overall accuracy of the model. However, the main constraint of this model is that it does not include and updating process or a recursive version that adds new data, unlike other traditional time-series approaches. The work presented by the authors in Ningkai Tang et al.[4] produces a forecasting model for the purpose of solar power generation prediction, based on least absolute shrinkage and selection operator (LASSO). An example of time sequential predictive models is presented in the work done by Chung-Hong Lee et.al[3]. The study is based on the scenario where the regular functioning of the field equipment and solar inverters facilitates the process of the entire power monitoring data being sent through the monitoring system. Based on the total sunshine intensity accumulated for a specific day and the generation capacity of the solar power plant, the efficiency of the plant is calculated to gain a better understanding of the climatic and environmental impacts on the system's ability to produce power. A statistical regression analysis on the dataset is performed and the mean absolute error and mean squared logarithmic error is recorded. While our proposed deep neural network fails to improve a relatively computationally heavy model in terms of mean squared logarithmic error, the performance in terms of mean absolute error is significantly better. The mean absolute error is a measure used for all methods in the proposed pipeline. The LSTM utilized in this paper has a mean absolute error that is marginally higher than the deep neural architecture from the proposed work.

A Khalyasmaa et.al[2] maes use of a random forest ensemble regression technique to make predictions of the power generated by the solar panel. The performance metrics are reported as mean absolute percentage error (MAPE). The MAPE for the approach presented in this paper is low when compared with that of the proposed methods. It does not provide a statistically significant

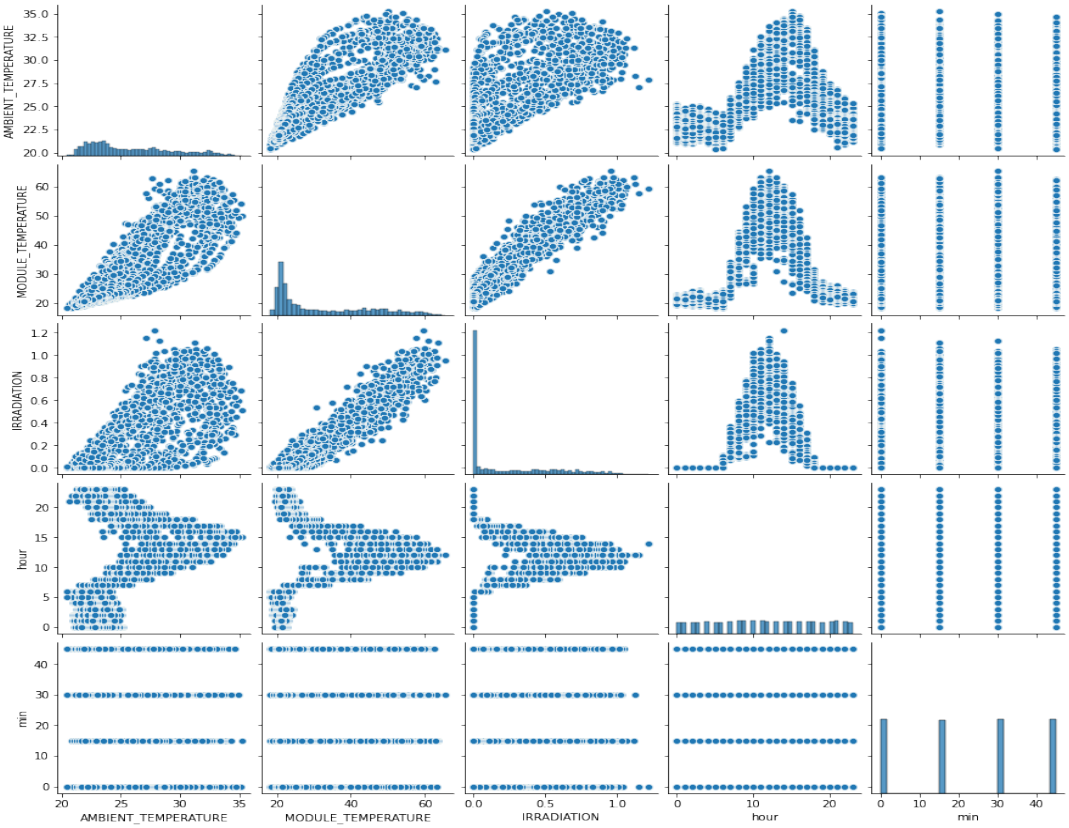

Figure 2: Pairwise Feature Plot of the Solar Power Generation Dataset

comparison of the ground truth to the predicted outcomes. While the comparison is challenging between the two methods, the statistical significance of the proposed method is proof that the used estimators are valuable. When compared based on mean absolute error, the one which reports a lower value can be used for prediction in the real-world scenario.

3 PROPOSED METHODS

The code for this work is available on Github[2]. The proposed pipeline is as seen in Figure 1. The dataset being used is a combination of two data sources containing sensor and power generation plant data by performing an inner join operation. The primary key used for the inner join is performed is the "DATE_TIME" attribute. The "AMBIENT_TEMPERATURE", "MODULE_TEMPERATURE" & "IRRADIATION" is obtained from the sensor dataset which contains these parameters recorded at 15 minute intervals throughout the day. Similarly, the power generation plant dataset contains the power generated across all inverters in the plant at 15 minute intervals throughout the day. The values of features ambient temperature, module temperature and irradiation are replicated as the plant has only one sensor for all inverters. The proposed methods focus only on predicting the DC_POWER because it is the direct output of the solar panel. In addition to this, the other parameters such as AC_POWER and the aggregation labels of DAILY_YIELD and TOTAL_YIELD can be derived from the predicted values of DC_POWER. To calculate the DC_POWER, a logarithmic scale is used primarily because it reduces the number of negative values returned by the neural network while making predictions. It is important to note that power generated at a negative value is the same as the power generated at a zero value. Better results have been observed for these experiments by utilizing a logarithmic scale. The date feature from the dataset has been rejected in this work as it possesses no impact on the amount of power generated.

Cyclical transformation is performed on the time of the day feature that has been extracted from the "DATE_TIME" variable due to its periodicity. The Feature Engine library is used for the encoding of cyclical features where two axes in a coordinate system are deployed. This is mathematically equivalent to a Fourier Transform of the signal. The time in minutes from 00:00 to 23:59 is transformed to represent the number of minutes that have passed since 00:00, thus making it a range of values from 0 to 1439. The difference in the time passed between 23:59 and 00:01 is calculated as 1339 minutes rather than 2 due to the cyclical nature of time. After this conversion of time to a range lying between 0 and 2π, or one cosine cycle, the cosine function is applied. The sine value of the corresponding cosine value is used to differentiate between the two cosine values that reoccur within the cycle.

[2]https://github.com/sumedha-98/CS228_DeepLearningProject

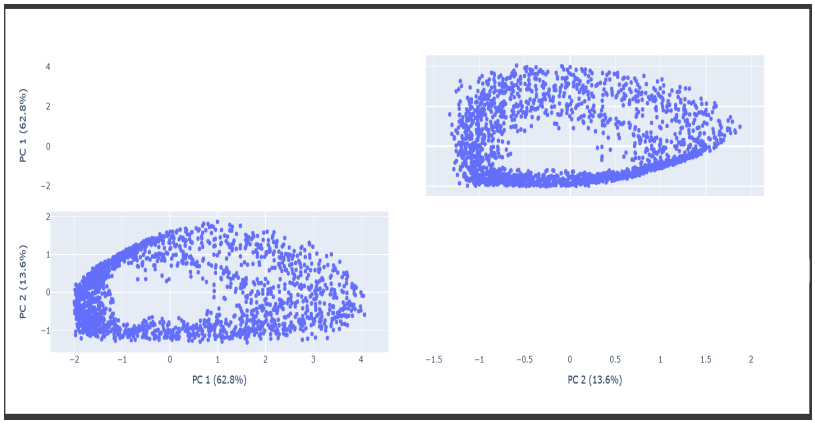

Figure 3: Principal Component Analysis of the Solar Power Generation Dataset

Principal Component Analysis is performed as seen in Figure 3. This is done to visualize the data and verify the feasibility of modelling it using the selected estimators. Another purpose of principal component analysis in this scenario is to check if any modifications are required to the estimators. The proposed approach utilizes a scatter plot to make sure that there are no strong correlations between the features and no discrepancies are present in predicting the dependent variable. The scatter plot is seen in Figure 2. The next step is to process the data which performs a z-score normalization on the ambient temperature, the module temperature and the irradiation of the solar panel. Along with this, the cyclical transformations on the hours and minute features are extracted from the date feature of the original dataset.

3.1 Gaussian Process Regression

Gaussian Process Regression[3] is an example of a supervised learning algorithm that is non-parametric in its nature[4]. It makes use of Gaussian Processes. Gaussian processes can be simply understood as a random group of variables that are selected in a manner where a subset of them would have Gaussian properties. This algorithm utilizes the covariance function in combination with a kernel. The kernel in this scenario is parameterized, which makes it have the abilitiy to make predictions on data. Gaussian process regression returns the distribution of probability on all functions that are capable of fitting the data that is provided. Similar to the Bayesian algorithm, Gaussian process regression selects a prior probability on the function space. It then proceeds to use the training data to compute a posterior probability. Further, it computes a predictive distribution on the testing data. Using these facts, the Gaussian process prior probability can be defined as:

$$f(x) \sim GP(m(x), k(x, x')) \tag{1}$$

where m(x) is the mean function and k(x, x') is the covariance function or the kernel function. For this implementation, a Radial Basis Function (RBF) has been chosen as the kernel function. The RBF kernel is of the form:

$$k(x, x') = e^{(-d(x,x')^2/2l^2)} \tag{2}$$

where 'l' is the scale of length of the kernel and function 'd' is the Euclidian distance. Due to the properties of the RBF kernel to encode, it is used in this work to improve the smoothing of the function. As the data used in this work does not have noise, it is important to include this component while computing the covariance matrix in the RBF kernel to recieve better results. The primary reason why it is important to include noise in the data is because the inherent property of regression is to produce a smooth signal by extracting from a noisy dataset[5]. Noise is some level of discontinuity incorporated between the diagonals of the computed covariance matrix. So, the kernel function can now be defined as:

$$K(x, x') = C(x, x') + \epsilon \tag{3}$$

where C(x, x') is the covariance matrix and ϵ is the noise component. The inclusion of the noise parameter ensures that the computations of the covariance matrix remain positive semi-definite.

3.2 Support Vector Regression

In the proposed implementation of the Support Vector Regression, every row of X and Y are iterated over a certain number of epochs. The weight update is done using the gradient descent algorithm. A bias term is used for further optimizing the results of the algorithm. In support vector regression, the selection of the support vectors is done in a manner where they lie within a specific decision boundary. The term used to ensure this is Epsilon, where for all values of X whose error value is lesser than the threshold value epsilon, the weights are updated using the Widrow-Hoff learning rule. This weight update rule uses the sum of the previous weight and the learning rate of the error and the features. To prevent overfitting, the validation set and checkpoint weight is used to facilitate early stopping. The checkpoint weight is the weight for which the validation loss is the least. Early stopping is implemented in the training loop such that it stops training after a certain number of epochs once the validation loss crosses a set tolerance level. To further

[3]https://gregorygundersen.com/blog/2019/09/12/practical-gp-regression/
[4]https://towardsdatascience.com/gaussian-process-regression-from-first-principles-833f4aa5f842

[5]https://bookdown.org/rbg/surrogates/chap5.html#chap5nugget

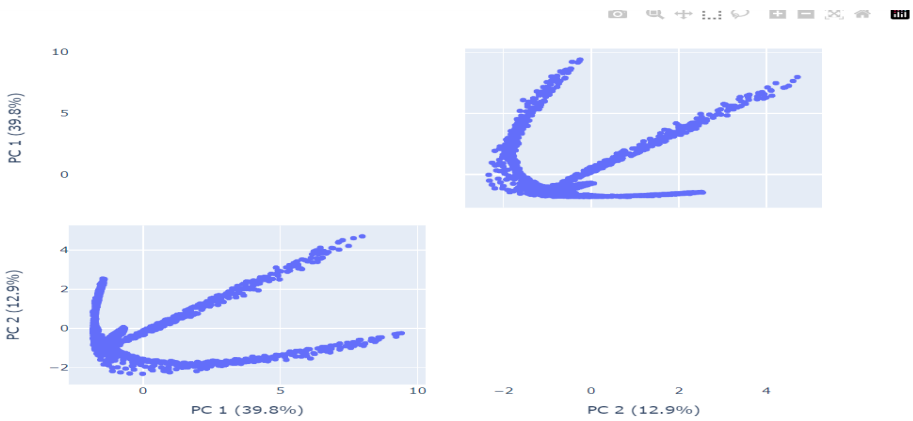

Figure 4: PCA with Polynomial Feature transform

improve the outcome of the algorithm, some regularization and feature representation methods are implemented.

(1) **Feature Representation :** t-SNE and Multi-dimensional Scaling (MDS) were implemented but does not operate on the dataset as expected due to the large amount of data present. The following two feature representations have been implemented and have provided conclusive results.
   (a) **Polynomial Feature Representation:** The scatterplot of the dataset shows the non-linear nature of the data. For the purpose of data transformation, the polynomial feature representation strategy is used. It transforms the input feature values into a new space where additional predictors are included. These new predictors are generated by raising the original predictors to a certain degree and by interactions of the features. After the implementation of the polynomial feature representation of degree 2, the resulting number of features is 29.
   (b) **Principal Component Analysis(PCA):** After polynomial feature representation, PCA is implemented on the 29 features to visualize the data. The comparison of support vector regression with PCA and without PCA is performed. The summary of the results is showcased in Table 2.

(2) **L1 and L2 Regularization:** L1 and L2 regularization has been implemented. The total of the absolute values of the weight is penalized by L1 regularization, and the sum of the weight's squares is penalized by L2.

3.3 Deep Neural Network Architecture

A Deep Multi-Layer Feed Forward Artificial Neural Network with 7 Hidden Layers is the pertinent method. The first layer, which has 40 neurons and uses the hyperbolic tangent ('tanh') as its activation function, is a dense layer. The initializer used for the layer is the Xavier Initialization with a L1-L2 regularization. As a Dropout Regularization layer, the second hidden layer randomly drops 3% of its neurons during each epoch. The next five hidden layers are dense layers, each having 30 neurons, 15 neurons, 10 neurons, and 4 neurons. 'Tanh' is employed as the activation function for the top layer and 'relu' for the next four levels. The last layer serves as

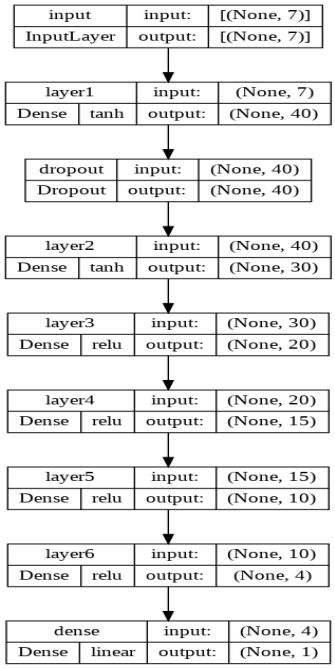

Figure 5: Deep Neural Network Architecture

the output layer and has a single linearly activated neuron because performing a regression job is the main goal. A batch size of 360 is used to train the algorithm, and the 'adam' optimizer, which makes use of adaptive momentum, is used for optimization[6].

The trained parameters were assessed using the mean_squared_logarithmic_error metric function, and the method used the mean_squared_error loss function. This decision was reached because, as stated in the aforementioned section 3, the dependent variable was evaluated on a logarithmic scale. The model summary is shown in Figure 5. The Figure 6 shows the loss plot for the proposed Deep

[6]https://arxiv.org/pdf/1412.6980.pdf

Neural Network Architecture. We eventually arrived at the proposed design after carrying out extensive experiments with various optimizers including 'adam', 'adagrad', 'adadelta', and 'sgd', exploring epoch values ranging from 30 to 100, and taking various layer sizes below 100 neurons as well as layer counts ranging from 2 to 8.

Implementations	Mean Absolute Error	T-Statistic	P-Value
Default off-the-shelf	0.231	0.602	0.547
Proposed off-the-shelf	0.143	-0.140	0.889
Proposed from scratch	0.147	-0.168	0.866

Table 1: Experimental Evaluation for GP Regression

Estimator	Mean Absolute Error	Root Mean Squared Error	P-value	T-statistic
Baseline	0.238	0.549	0.911	0.111
From scratch(without PCA)	0.822	1.093	0.845	-0.916
From scratch(with PCA)	0.860	1.150	0.926	-0.093

Table 2: Measures of Comparison for SVR

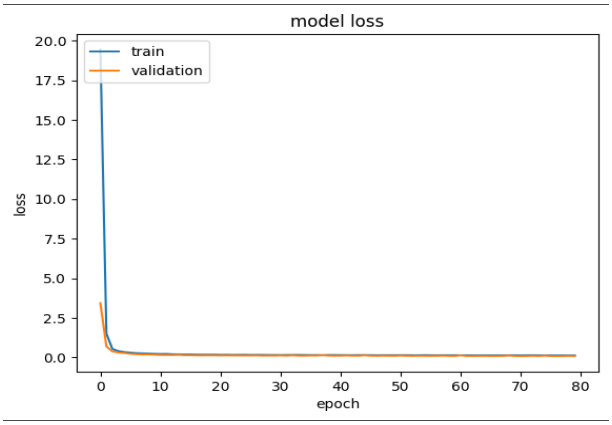

Figure 6: Deep Neural Network Architecture Train vs Validation Loss Plot

We tested the early halting callback technique using 5, 8, and 10 epochs of patience with various patience levels. However, we discovered that the most efficient method for choosing the proper parametric weights for the neurons was to save the optimal weights throughout a maximum number of epochs.

The comparison of the statistical significance of the result is done by performing a two-tailed Student-T Test. As we can see from Table 4 the two-tailed Student-T test for the predicted and ground truth samples returns P-Values which are much larger than 0.025(for 95% confidence), thereby showing that we cannot reject the null hypothesis and that the means of the sample and the population are equal. Hence our model produces statistically significant results for the task at hand.

4 EXPERIMENTAL EVALUATION

4.1 Gaussian Process Regression

Table 1 describes the results of the models implemented using the default settings from the off-the-shelf library, the proposed method using the off-the-shelf library and finally, the model implemented from scratch. These results are in terms of Mean Absolute Error, T-Statistic and P-Value.

4.2 Support Vector Regression Results

Table 2 shows the results of the model implemented using the off-the-shelf library and the model implemented from scratch.

4.3 Deep Neural Network Architecture Results

Table 3 shows the cross-validated results of the proposed Deep Neural Network and Table 4 shows the statistical significance of the estimator against the ground truth values.

5 DISCUSSION & CONCLUSIONS

The paper performs an extensively exhaustive comparison of Gaussian Process Regression, Support Vector Regression and the proposed Deep Neural Network Architecture for Solar Power Generation Prediction data. According to the results, the Deep Neural Network demonstrates superior performance compared to the other methods, exhibiting a significant margin with a Mean Absolute Error of 0.0726 and a Standard Deviation of 0.01 in a 5-fold cross-validation setting.

The outcomes unequivocally show off neural networks' remarkable function estimation skills. Support Vector Regression is inferior to the Deep Neural Network despite polynomial modification of the features to improve linear decision boundary prediction. On over 50% of the data, Gaussian Process Regression, however, performs quite well because to its accurate calculation and cost function optimization. However, the processing costs of the straight linear algebraic calculation offer difficulties, making the viability of inference challenging. Additionally, Gaussian Process Regression is unsuitable for common application since it requires inverting the covariance matrix using the cholesky transform in order to forecast a single data point. The Deep Neural Network, on the other hand, stands out since it does not require a polynomial feature transformation prior to modeling and outperforms other approaches considerably in terms of Mean Squared Logarithmic Error (MSLE). The MSLE metric is a crucial comparison tool since the dependent variable is predicted using a logarithmic scale. With an average P-Value of 0.866 versus the ground truth prediction population, the suggested technique also yields statistically significant results.

Although the results are statistically significant and the inference time is reasonably small, the suggested technique admits that there is a substantial possibility to improve the feature set by including weather data and other variables. This is due to the fact that the solar panel's sensor captures diffused irradiation, which frequently results in predicted power levels that are a little bit greater than the power actually produced by the panel. Consequently, the real conversion of power to AC_POWER at ground level may be somewhat reduced, despite the fact that the inaccuracy in result comparison may look lesser. Without relying on the status of the solar panel

Estimator	Seed	Mean Absolute Error	Root Mean Squared Error	Mean Squared Logarithmic Error	Mean ± SD (MAE)
Sci-Kit Learn Off The Shelf Baseline	40	0.0780	0.320	0.0126	**0.0947 ± 0.015 [0.080, 0.101]**
	41	0.0972	0.341	0.0132	
	42	0.0806	0.294	0.0123	
	43	0.1108	0.384	0.0136	
	44	0.1070	0.265	0.0094	
Proposed Approach Implementation From Scratch	40	0.0669	0.304	0.0109	**0.0726 ± 0.0100 [0.0626, 0.0826]**
	41	0.0886	0.320	0.0108	
	42	0.0710	0.290	0.0103	
	43	0.0742	0.340	0.0107	
	44	**0.0622**	**0.232**	**0.0062**	

Table 3: Cross Validation Results with varied Random Seeds for Deep Neural Network Architecture

Algorithm	Mean Absolute Error	Root Mean Squared Error	Mean Squared Logarithmic Error	T-Statistic	P-Value
Sci-Kit Learn Baseline	0.0806	0.294	0.0123	0.111	0.912
Proposed Approach	0.0710	0.290	0.0103	0.169	0.866

Table 4: Measures of Comparison for Deep Neural Network Architecture

as determined by the ambient sensor, the model may also predict future values with accuracy by including meteorological data. This improves panel maintenance and the predictive capabilities.

6 REFERENCES

[1] Mariam AlKandari and Imtiaz Ahmad. "Solar power generation forecasting using ensemble approach based on deep learning and statistical methods". In: *Applied Computing and Informatics* (2020).

[2] Alexandra Khalyasmaa et al. "Prediction of Solar Power Generation Based on Random Forest Regressor Model". In: *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*. IEEE. 2019, pp. 0780–0785.

[3] Chung-Hong Lee, Hsin-Chang Yang, and Guan-Bo Ye. "Predicting the Performance of Solar Power Generation Using Deep Learning Methods". In: *Applied Sciences* 11.15 (2021), p. 6887.

[4] Ningkai Tang et al. "Solar power generation forecasting with a LASSO-based approach". In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 1090–1099.