CS235 Fall'22 Project Final Report: Comparison of Statistical Data Mining Techniques & Deep Learning Methodologies for Solar Power Generation Prediction

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

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

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

Naveen Devadi
University Of California, Riverside
Computer Science
862395113
ndeva005@ucr.edu

Tejas Milind Deshpande
University Of California, Riverside
Computer Science
862393211
tdesh006@ucr.edu

ABSTRACT

With the fossil fuel reserves depleting at an alarming rate and energy scientists transitioning to renewable sources of energy, it is essential to ensure that technology moves with the times and the systems being used are efficiently maintained. The overwhelming transition to renewable sources of energy to generate electricity for daily use is rapidly transforming the landscape of the electric power generation market spectrum. This paper presents a comparison of the best methodologies that can be adopted for predictive maintenance of a solar panel. The proposed pipeline uses input data from the sensor placed in a solar power generation plant to predict the power generated by the solar panel at any time of the day. This provides efficient solar panel maintenance since any discrepancies observed between the recorded values of the power generated and the values predicted by the best estimator will allow the administrative staff at the power plant to understand that a particular inverter is malfunctioning.

KEYWORDS

Solar Power Generation, Statistical Data Mining, Deep Learning, Gaussian Process Regression, Support Vector Regression, Random Sample Consensus (RANSAC), Least Absolute Shrinkage and Selection Operation (LASSO), Neural Networks

1 INTRODUCTION

The benefits of solar energy as we know are innumerable, especially in the field of generating electricity. Solar power is a pollution-free, renewable source of electricity which makes it highly significant in today's day and age. Even with its vast benefits, solar power accounts for just 3.6% of the power generated across the globe due to the inefficiencies in the machinery required by power plants to generate solar energy. Predicting the generation of solar power plays a crucial role in increasing efficiency, thereby increasing its production. Estimating the amount of power generated will lead to improved grid management, efficient solar panel maintenance and quicker identification of faulty equipment in the system.

Harnessing solar energy is not very easy because of various factors like conversion rate of solar radiation to electricity, weather conditions, landscapes, cost and maintenance of equipment, etc. In order to optimize the process of harnessing solar energy while ensuring systemic efficiency and reduced maintenance costs, monitoring and deriving patterns from past data is highly essential. Using strong data mining techniques and statistical analysis, the prediction of power generated on a daily basis by a solar power plant will benefit the consumers and equipment manufacturers alike since any kind of mishaps can be mitigated and maintenance and solutions can be planned ahead of time. This paper aims to compare various data-driven estimators and arrive at a conclusive methodology that can be followed with the least possible erroneous technique so that fewer resources are wasted and coherence and productivity are maintained.

The proposed approach uses the dataset titled Solar Power Generation[1] which contains data from two solar power plants for a period of 34 days. It has two pairs of files, each pair having data about the amount of power generated by the plant and a dataset having sensor readings for the day. The sensor data which is gathered at a plant level consists of the ambient temperature, module temperature, and irradiation. The dataset, which is gathered at the inverter level where each inverter has multiple lines of solar panels attached to it, consists of the amount of DC Power and AC Power generated in 15 minute intervals and the daily yield and total yield of the plant. The daily yield is a cumulative sum of the power generated on that day until that point in time. The total yield is a cumulative sum of the power generated by the inverter until that day at that point in time. The paper is organized in the following way in the subsequent sections:

CS235 F22, DMT,

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

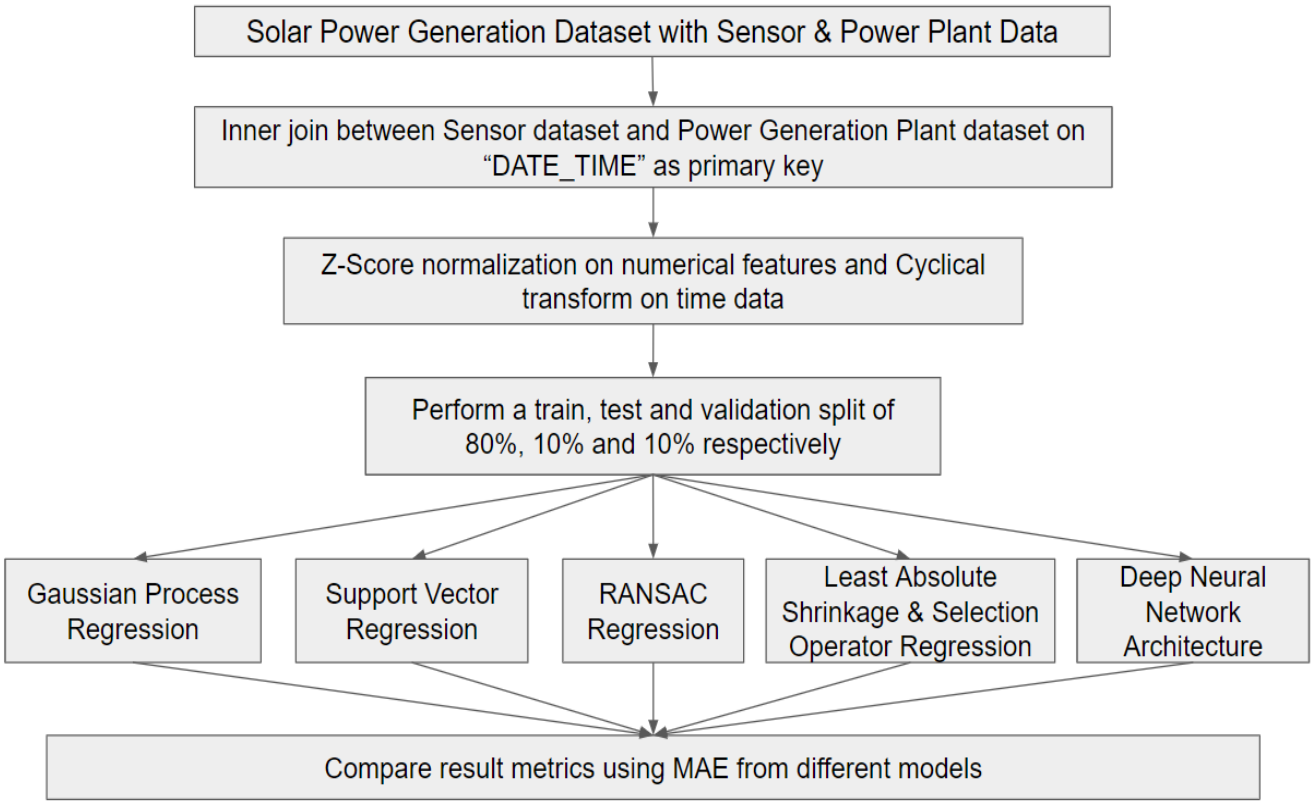

Figure 1: Project Pipeline Overview

(1) Related Work compares the existing methodologies on Solar Power Generation Prediction
(2) The Proposed Methods section compares and contrasts the 4 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.

2 RELATED WORK

The related work for Solar Power Generation Prediction is extensive and the literature covers different kinds of problem statements that apply Deep Learning based solutions like Long Short-Term Memory (LSTM)[1], Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU) and Time Sequential Predictive Models[3] among others.

Mariam AlKandari et.al[1] predicts the solar power that will be generated a day ahead by the solar panel given the weather forecast and the condition of the solar panel on the current day. While this problem statement seems slightly different from our approach since these predictions would be used more for preventive maintenance rather of the solar panel than for verifying if a particular solar panel is malfunctioning, the paper still reports mean absolute error among one of its metrics and hence we choose to compare against this methodology. This paper proposes an ensemble based constrained GBRT model of regression trees. The GBRT model has been expanded to include regression while being primarily developed for classification. In addition, GBRT is a machine learning model that improves results by combining the output of numerous tiny regression trees of a defined size. The proposed model adds a significant constraint because, unlike the traditional time series approaches, it doesn't include an updating process or recursive version as new data are added. A forecasting model for solar power generation based on the least absolute shrinkage and selection operator (LASSO) was proposed by the authors in Ningkai Tang et al.[4]. Chung-Hong Lee et.al[3] uses time sequential predictive models. The whole power monitoring data of the solar power station is received through the monitoring system when the solar inverter and field equipment are functioning regularly, which is the scenario on which this study is based. Then, based on the total amount of sunshine intensity accumulated on that day and the power plant's

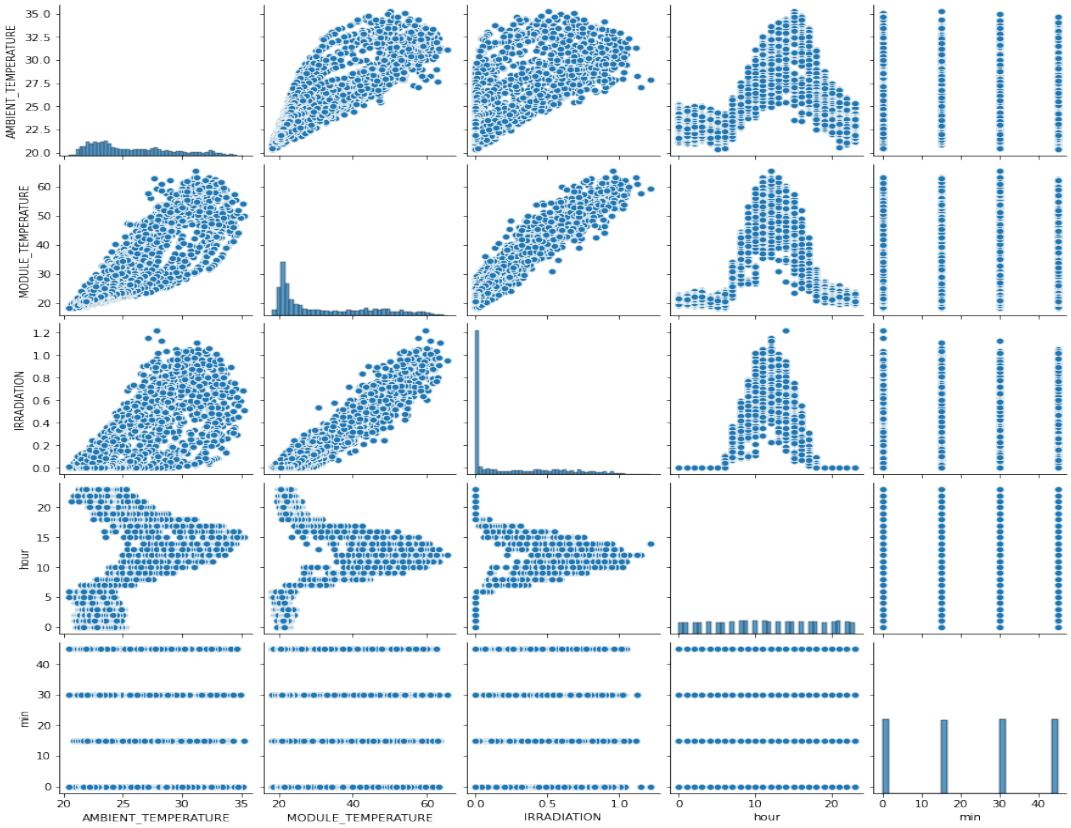

Figure 2: Pairwise Feature Plot of the Solar Power Generation Dataset

power generation capacity, it calculates the efficiency of the solar power system to better understand how climate and environmental factors affect the solar system's ability to generate power. Chung-Hong Lee et.al[3] performs a statistical regression analysis on the dataset used in the study and report the mean absolute error and mean squared logarithmic error. While our proposed Deep Neural Network fails to better a relatively computationally heavier model in the mean squared logarithmic error, it reports a much better performance in terms of mean absolute error, the metric we use to evaluate all of our methods in the pipeline. The LSTM used in this paper observes a mean absolute error marginally higher than the Deep Learning Architecture of the proposed pipeline.

A Khalyasmaa et.al[2] uses a Random Forest ensemble regressor to predict the power generation outcome of the solar panel. The reported metrics are in the form of Mean Absolute Percentage Error (MAPE). The MAPE for the approach in the paper is low but does not provide a statistically significant comparison of the ground truth to the predicted outcomes unlike the proposed pipeline in our approach. While the comparison is difficult to obtain between the two methods, statistical significance of our proposed approach tells us that the estimators used are valuable and when compared with each other on the mean absolute error, we can choose the one which reports the least value for prediction in a real life scenario.

3 PROPOSED METHODS

The code for this paper is available on GitHub[2]. The pipeline for the proposed approach is as shown in Figure 1. The final dataset is curated from two data sources by performing an inner join on the Sensor and Power Generation Plant Data with "DATE_TIME as the primary key on which the join is performed for both tables. The Sensor dataset records "AMBIENT_TEMPERATURE", "MODULE_TEMPERATURE" & "IRRADIATION" for 15 minute intervals throughout the day. The Power Generation Plant Dataset records the power generated across all the inverters in 15 minute intervals throughout the day. Since the plant only has one sensor for all inverters, we replicate the feature values (ambient temperature, module temperature and irradiation) for each of the inverters across the dataset. The proposed pipeline only focuses on predicting the DC_POWER since that is the direct output of the solar panel. The conversion to AC_POWER and aggregation labels of DAILY_YIELD and TOTAL_YIELD can then be derived from the predicted DC_POWER. We use the logarithmic scale to calculate the DC POWER. This decreases the number of negative values the neural network returns when predicting DC POWER, which we do not require because power generated at a negative value is the same as power created at a zero value. Since it produces better results for our experiments, the logarithmic scale is more appropriate. We also

[2]https://github.com/MannyBoink/CS235_Data_Mining_Solar_Power

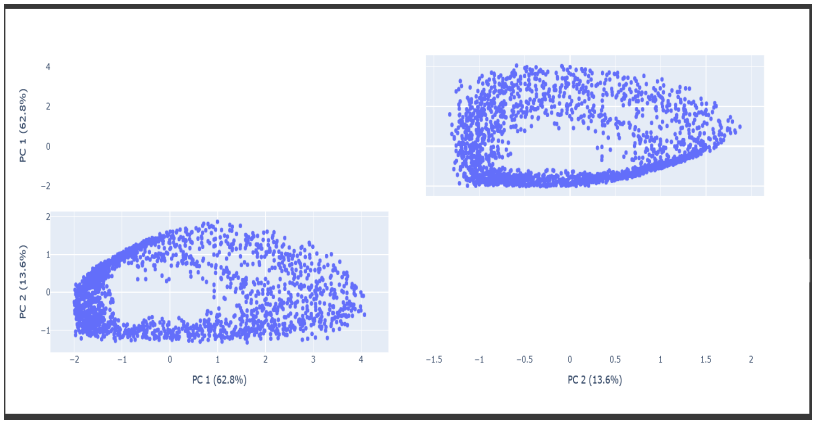

Figure 3: Principal Component Analysis of the Solar Power Generation Dataset

reject the date feature because the date the reading was obtained has no bearing on the amount of power generated.

The proposed approach performs a cyclical transformation on the time of day feature extracted from the "DATE_TIME" variable in the dataset due to its periodic nature. Two axes are used in a coordinate system for the cyclical feature encoding from the Feature Engine library. The time in minutes between 00:00 and 23:59 is represented by the number of minutes that have passed since 00:00. The range of values for this is 0 to 1439. The difference in time between 23:59 and 00:01 is calculated as 1339 minutes rather than 2 due to the cyclical structure of time. After converting time to a range between 0 and 2π, or one cosine cycle, the cosine function is applied. We use the sine value of the corresponding cosine to distinguish between the two cosine values that reoccur in the cycle.

Principal Component Analysis is performed as shown in Figure 3 for Data Visualization in order to verify the feasibility of the data and whether it can be modeled using the estimators in certain methods and if there are any modifications required. The proposed approach also uses a scatter plot to ensure that there are no discrepancies or strong correlations among the features being used for predicting the dependent variable as shown in Figure 2. The data processing stage of the proposed pipeline then performs z-score normalization on the ambient temperature, module temperature and irradiation of the solar panel and cyclical transformations on the hour and minute features extracted from the date feature of the initial dataset.

3.1 Gaussian Process Regression

Gaussian Process Regression is a supervised learning algorithm and is non-parametric in nature[3]. It utilizes Gaussian Processes which are essentially a random group of variables selected in a manner where a subset of them has Gaussian properties. The algorithm makes use of a covariance function in conjunction with a kernel that is parameterized in order to make predictions. Gaussian process regression calculates the distribution of probability on all functions that are capable of fitting the data provided. Much like the Bayesian algorithm, GPR selects a prior on the function space, uses the training data to compute the posterior and then the predictive distribution on the testing data. Using this information, we can define a Gaussian Process Prior 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. The purpose of this kernel is to improve the smoothing of the function by encoding. Since the data in question has no noise, it is important to include this component while computing the covariance matrix in the RBF kernel. This is mainly attributed to the fact that regression is the process of extracting from noisy data and producing a smooth signal[4]. The noise is essentially 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 implemented algorithm for the Support Vector Regressor, we iterate through every row of X and Y of the dataset "Epoch" number of times. To update the weights we make use of Gradient descent and by making use of bias we optimize the algorithm further. In Support Vector Regression, the support vectors are selected such that they lie within the decision boundary. This is ensured by making use of Epsilon. For all the values of X whose error of predicted value is lesser than the Epsilon, the weights are updated using the Widrow-Hoff Learning Rule. In Widrow-Hoff Learning Rule, the weight to be updated is equal to the sum of the old weight

[3]https://towardsdatascience.com/gaussian-process-regression-from-first-principles-833f4aa5f842

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

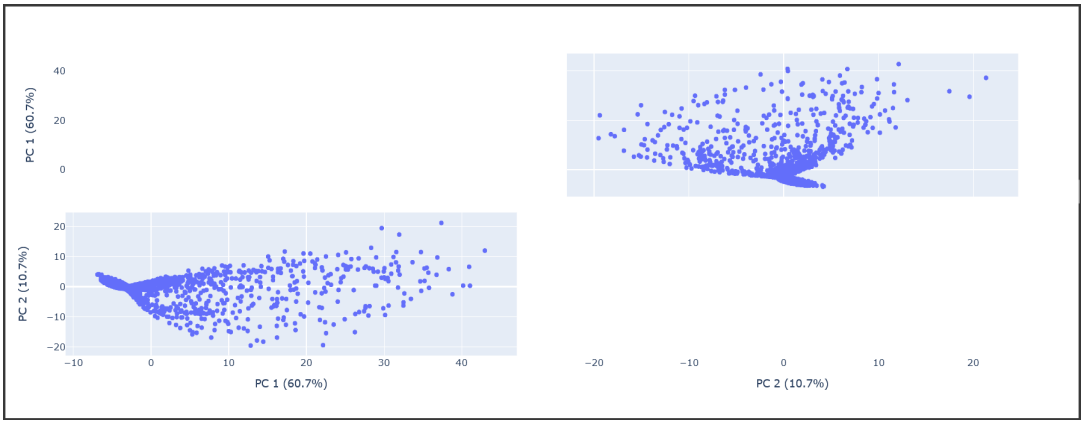

Figure 4: PCA with Polynomial Feature transform

and the learning rate of the error and the features. In order to avoid overfitting, early stopping is implemented by making use of the validation set and checkpoint weight. The checkpoint weight is equal to the weight for which the validation loss is minimum. Early stopping is implemented by setting a tolerance for Epoch such that the Epoch loop is stopped once the validation loss crosses the set tolerance. Some Feature Representation and Regularization methods have been implemented to improvise on the algorithm.

(1) **Feature Representation :** T-sne and Multi-dimensional Scaling(MDS) were implemented but does not work on the dataset due to the large amount of data present. The following two feature representations have been implemented sucessfully.
   (a) **Polynomial Feature Representation :** The dataset scatterplot shows that the dataset is non-linear in nature. In order to transform the data, we use Polynomial Feature Representation which transforms the input feature values to a new space. In this new space, extra predictors are added, which are generated by raising the original predictors to a power and by interactions of the features. Post implementation of Polynomial Feature Representation of degree 2, we get 29 features.
   (b) **Principal Component Analysis(PCA) :** PCA is implemented post Polynomial Feature Representation for 29 features. The dimensionality of the dataset is reduced to 20 components with PCA. We have implemented the SVR model with and without PCA components and the results are showcased in Table 5.

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

3.3 Random Sample Consensus (RANSAC)

One of the methods used to predict the DC power is Random Sample Consensus (RANSAC) regression. We have used Linear Regression as the base estimator function in our RANSAC implementation. RANSAC algorithm makes the Linear Regression model more robust to outliers, by training the Linear Regression model multiple times. It starts by training the Linear Regression on randomly selected minimum data points and then predicts the DC power generated for remaining data points(validation set). The validation set samples for which prediction error value is less than specified threshold are combined with the initial minimum samples.

If the number of inliers are greater than the specified stop inliers, then the model is retrained using these combined data points and scored using mean absolute error (MAE). If the MAE for the current model is less than the best prior model, then record / update the current model as the best model and the model MAE as the best error. This is done for a specified number of iterations and finally the best found model is returned. Before executing the RANSAC Regression, we have transformed the input features using Polynomial Features.

The RANSAC regression algorithm removes the outliers, but it still uses linear regression as the base estimator, and our dataset does not contain linear data, hence we used polynomial features to transform the input feature values to a new space. In this new space, extra predictors are added, which are generated by raising the original predictors to a power and by interactions of the features. PCA visualization of input features after polynomial transform is shown in Figure 4.

Along with polynomial features, we also experimented with other feature representations such as Power Transformer, Principal Component Analysis, Factor analysis and Spline Transformer. The features generated by Polynomial Features seem to provide the best prediction performance out of all the five representations.

Apart from Polynomial Features, Spline Transformer features show better prediction performance than the remaining three feature representations. This may be because Spline Transformer generates extra features similar to Polynomial Features, by raising the original features to the specified degree, with the exception of interaction terms.

Thus, generating additional features based on higher degrees and interactions among original features seems to generate better input features for RANSAC model with Linear Regression base estimator instead of reducing features using dimensionality reduction techniques.

3.4 Least Absolute Shrinkage and Selection Operator (LASSO)

The algorithm of choice for implementation for the prediction of DC Power is LASSO regression. The acronym "LASSO" stands for Least Absolute Shrinkage and Selection Operator. This is used for better and more accurate predictions compared to other regression methods. This model makes use of shrinkage where the data values are accumulated at a central point such as the mean. LASSO regression performs L1 regularization by adding a penalty equal to the absolute value of the coefficient magnitude. This type of regularization can lead to simpler models with fewer coefficients. Some coefficients may go to zero and be removed from the model. Higher penalties consequentially centres the coefficient values proximal to zero and are most preferred for producing simpler models. At the implementation level, during gradient descent optimization, the added penalty diminishes the weights closer to zero or zero. The weights which are dwindled to zero eradicate the features present in the hypothetical function. Therefore, immaterial features don't take part in the predictive model. This penalization of weights renders the hypothesis straightforward, which uplifts the simplicity of the model. Although primitively defined for linear regression, LASSO regularization can be smoothly extended to other statistical models. LASSO's ability to perform subset selection depends on the form of the constraint and has various interpretations. Linear regression gives the regression coefficients observed in the dataset. Using LASSO regression, we can narrow down or normalize these coefficients to avoid overfitting and work better on different data sets. LASSO regression penalizes less important features of the data set and eliminates them and therefore, has the advantage of making feature selection and model creation easier.

StandardScaler is leveraged to scale the data which helps to resize the distribution of values such that the mean of the observed values is 0 and the standard deviation is 1. Taking into account that the data is in a time series fashion, cyclical transform is used on the DateTime component which improves the modeling process by augmenting the information provided to the algorithm.

3.5 Deep Neural Network Architecture

The implemented algorithm is a Deep Multi-Layer Feed Forward Artificial Neural Network with 7 Hidden Layers. The first layer is a Dense Layer containing 40 neurons with 'tanh' as the activation function. The second hidden layer is a Dropout Regularization layer with 3% of neurons being dropped out at each epoch at random. The next 5 hidden layers are dense layers of size 30, 20, 15, 10 and 4 neurons with activations of'tanh' for the first and 'relu' for the remaining 4 layers. The final layer is a single neuron output with linear activation as we are performing a regression task. The proposed algorithm uses a batch_size of 360 and the 'adam' optimizer for adaptive momentum[5]. The loss function used for the algorithm was the 'mean_squared_error' and the metric function used to validate the trained parameters was the 'mean_squared_logarithmic_error' function since we evaluate the dependent variable on a logarithmic scale as mentioned in Section 3. The model summary is shown in Figure 5. The loss plot for the proposed Deep Neural Network Architecture is shown in Figure 6. We arrive at the proposed architecture

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

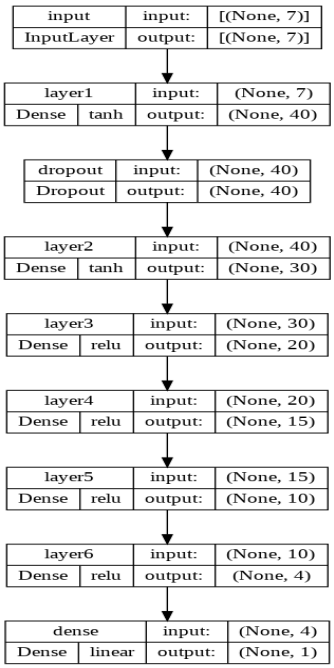

Figure 5: Deep Neural Network Architecture

after extensive experimentation with 'adam', 'adagrad', 'adadelta', and 'sgd' as the optimizers, various epoch values ranging between 30 and 100, multiple layer sizes less than 100 neurons and number of layers ranging from 2 to 8. The proposed architecture is finalized

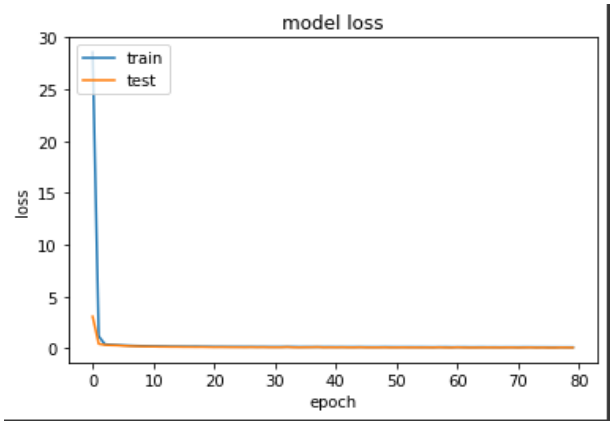

Figure 6: Deep Neural Network Architecture Train vs Validation Loss Plot

after extensive experimentation with 'adam', 'adagrad', 'adadelta', and 'sgd' as the optimizers, various epoch values ranging between 30 and 100, multiple layer sizes less than 100 neurons and number of layers ranging from 2 to 8. The early stopping callback mechanism was also experimented with a patience of 5, 8 and 10 epochs but saving the best weight over a maximum number of epochs was eventually the best way of choosing the correct parametric weights for the neurons.

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 1: 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 2: Measures of Comparison for Deep Neural Network Architecture

Metric	LASSO Baseline	LASSO from scratch
RMSE and Standard Deviation	523.065 (SD: 43.179)	507.720 (SD: 129.41)
MAE and Standard Deviation	261.701(SD: 6.685)	262.232 (SD: 41.083)
R2 Score and Standard Deviation	0.983 (SD: 0.002)	0.982 (SD: 0.010)
T-Statistic	-0.178	-0.039
P-Value	0.859	0.679

Table 3: Experimental Evaluation for Least Absolute Shrinkage and Selection Operator (LASSO)

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 4: Experimental Evaluation for Gaussian Process 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 5: Measures of Comparison for SVR

Implementations	T-Statistic	P-Value	Mean Absolute Error(MAE)	Root Mean Squared Error (RSME)
Baseline without polynomial features	0.854	0.393	1.103	1.971
Baseline with polynomial features	0.044	0.965	0.474	0.529
From Scratch Implementation without polynomial features	0.419	0.675	1.102	1.964
From Scratch Implementation with polynomial features	-0.018	0.986	0.500	0.602

Table 6: RANSAC Experiment Results Comparison

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 2 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 two populations are equal. Hence our model produces statistically significant results for the task at hand.

4 EXPERIMENTAL EVALUATION

4.1 Gaussian Process Regression

Table 4 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 5 shows the results of the model implemented using the off-the-shelf library and the model implemented from scratch.

4.3 Experiment Results for Random Sample Consensus(RANSAC)

Table 6 summarizes RANSAC regression performance with and without Polynomial Features for baseline model and implementation from scratch.

4.4 Least Absolute Shrinkage and Selection Operator (LASSO)

Table 3 shows the results of the model implemented using the off-the-shelf library and the model implemented from scratch

4.5 Deep Neural Network Architecture Results

Table 1 shows the cross-validated results of the proposed Deep Neural Network and Table 2 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, RANSAC Regression, LASSO Regression and the proposed Deep Neural Network Architecture for Solar Power Generation Prediction data. As the results show, the Deep Neural Network outperforms the other methods by a large margin with a 5-fold cross validation Mean Absolute Error of 0.0726 and a Standard Deviation of 0.01.

The power of function estimation of Neural Networks is clearly depicted in the results. Methods such as RANSAC and Support Vector Regression are not able to perform as well as the Deep Neural Network even after the features have been polynomially transformed into a better feature space for linear decision boundary prediction. Gaussian Process Regression on the other hand, is an exact computation and calculation of the optimization problem on the cost function and hence performs extremely well on nearly 50% of the data. The direct linear algebraic calculation does not permit inference feasibility since inversion of matrices is computationally expensive. Prediction on just a single data point also requires the Gaussian Process Regression to invert the covariance matrix using the cholesky transform and is not practical for daily use. The Deep Neural Network not only does not require a polynomial feature transformation prior to modelling but also outperforms the other methods significantly in the Mean Squared Logarithmic Error (MSLE). Since the dependent variable is predicted on the logarithmic scale, the MSLE metric is an important measure of comparison. Finally, the proposed approach shows that the results of the estimator are statistically significant with the average P-Value against the ground truth prediction population being 0.866.

The proposed approach observes that even though the inference time is relatively low and that the results are significant, there is strong scope of incorporating weather data and other factors in the feature set since the irradiation being recorded by the sensor on the solar panel is diffused irradiation. The diffused irradiation data makes it so that the predicted power would normally be slightly higher than the actual power being generated by the panel. Thus, even though the result comparison may show less error, the actual ground level conversion to AC_POWER will be slightly reduced. With the weather data being incorporated, the model can also predict accurate future values without the presence of the current state of the solar panel from the ambient sensor, thereby increasing predictive power and improving panel maintenance.

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.