Short-term solar power forecasting using different Machine Learning models

Diego Rueda Uribe
Department of Mechanical Engineering
Universidad de los Andes
Bogotá, Colombia
d.rueda@uniandes.edu.co
June 2020

Advisor: Andrés Leonardo González Mancera, PhD.

Presented to obtain the degree of:

BSc. Mechanical Engineer

Abstract – **Photovoltaic power production forecasting has become increasingly relevant over the past decade. In this study, training and evaluation of different Machine Learning models is performed in order to generate short-term predictions based on current power production and weather variables including temperature, relative humidity, wind speed and direction, cloud cover, and direct radiation. Three different datasets were used to train each model, from three different geographic locations in the world.**

I. Introduction

Over the past decade, the world has seen a considerable increase in the use of renewable energy due to ever declining costs and as part of a global effort to reduce greenhouse gas emissions. The share of renewable energy in total global energy production has risen from 0.7% in 2010 to 2% in 2017. Photovoltaic (PV) installed capacity has grown from 21 GW in 2009 to 505 GW in 2018, nearly a 23-fold increase in just nine years. This trend has occurred mainly because the levelized cost of energy (LCOE) of a PV installation has rapidly declined over the years. Globally, the average LCOE for PV systems has declined 77% between 2010 and 2018 [1]. As costs decline and new projects are being made, industrial and academic interest on PV energy has risen accordingly.

However, PV energy production is unpredictable because it depends on the amount of solar irradiance on the panels, amongst other uncontrollable factors such as temperature, wind speed, relative humidity, among others, further analyzed in this work. The inherent unpredictability associated to PV energy production has raised the industry's interest to further develop accurate forecasting methods. Forecasting PV power is relevant because grid operators must be able to accurately know electric energy production and try to match it with electricity demand. As renewables take on a larger share of the electricity production, matching supply with demand has been an increasingly challenging and relevant task [2].

Solar power forecasting for PV system design has been traditionally based on physical models that depend on irradiance measurements and the geographic position of PV modules relative to the sun's movement [3]. However, different Machine Learning models have been recently introduced for PV power forecasting mainly in two different ways: to predict global horizontal irradiance or to predict PV power directly. Global horizontal irradiance predictions are then fed to a PV physical model that considers system characteristics such as module nominal power, inclination, inverter efficiency, among others. This approach does not necessarily require historic power data to work and is commonly used to evaluate whether a particular site is adequate for a PV power plant or not. On the other hand, forecasting PV power directly with Machine Learning depends on historic power data of an existing PV installation. Both types of approaches rely heavily on weather data availability and have proven to be increasingly accurate. For example, authors in [4] develop a framework to find an optimal Machine Learning model that forecasts GHI for a particular measuring station in Abu Dhabi International Airport, UAE. The study in Abu Dhabi finds that models based on a Bayesian framework result in an average determination coefficient R^2 of 0.969. An example of direct PV power production found in [5] compares different Deep Learning and statistical models on ten different datasets from PV installations in South Korea. Results show that Deep Learning models combining Convolutional Neural Networks (CNN) and Long Short-Term Memory Networks (LSTM) outperform other models. Also, the study made in South Korea demonstrates that including weather data as an input for the models improves performance.

In this paper, different Machine Learning techniques are implemented to forecast 5 or 10 minute-ahead PV power for three different systems, based on current power production and weather-associated variables. The first step in this study was data preprocessing and handling (i.e. eliminating null values, joining data from different sources, among others). Second, a Principal Component Analysis (PCA) was performed on the data in order to extract the most relevant data features to train the Machine Learning models. Third, different Machine Learning and Deep Learning models are trained and tested. Finally, results are discussed, the best model is identified, and all models are compared.

II. Data Handling

PV Systems

The three PV power installations analyzed in this project are located in Perth (Australia), Chapman (Australia), and Bogotá (Colombia). System characteristics are shown in table I.

Table I. PV system characteristics.

Location	**Latitude**	**Longitude**	**Elevation above sea level [m]**	**Installed capacity [kWp]**	**Module reference**	**AC Inverter reference**
Perth	-31.929137	115.913199	30	3.04	N/A	N/A
Chapman	-35.355794	149.047986	645	10.26	N/A	N/A
Bogotá	4.604303	-74.065887	2600	80.06	200 x LG Electronics LG400N2W-A5	1 x PVS-50-TL 1 x TRIO-27.6-TL-OUTD

Data sources:

Power data for the three selected locations was obtained from two different sources: the data for two locations (Chapman and Perth) was downloaded from an online public database of thousands of individual PV systems across three regions in Australia available in [6], while the third location's power data was directly downloaded from an online platform built by meteocontrol GmbH for the solar panels that are on the roof of the SD block (*Bloque Julio Mario Santodomingo*) at Universidad de los Andes in Bogotá, Colombia.

The database from which power data was downloaded contains a file for each region in Australia (Perth, Canberra and Adelaide) with a timeseries of 10-minute average solar power for thousands of PV systems. These values are normalized, so they must be multiplied by each system's installed capacity to obtain the actual power value. Another file in the database contains general information about each PV system such as ID, geographic position, installed capacity, among others. The two locations from the first source of Australian PV systems were selected because they presented few missing values and therefore they offer a more complete representation of a PV power plant with normal, uninterrupted operation. Both locations seem to be residential units, according to satellite imagery from Google Maps. For the scope of this project, it is not possible to physically verify these systems. Data has a 10-minute temporal resolution and was filtered using quality control techniques detailed in [6]. The data spans from September 23th, 2016 to November 30th, 2017.

On the other hand, power data from the third location in Bogotá is directly measured and available for analysis online for educational purposes within Universidad de los Andes. Data has a 5-minute temporal resolution and is available between August 3rd, 2019 and May 9th, 2020. In this case, only power readings were downloaded.

Figure 1. Satellite Image of PV System in Chapman, Australia obtained from Google Maps.

Figure 2. PV installation at SD Building. Bogotá, Colombia

Weather data was downloaded from meteoblue history+, which is a global weather simulation archive that has hourly data for many weather variables since 1985. This organization supported this research project by providing this data free of charge. The variables used for this project are: Temperature (2 meters above ground), Relative Humidity (2 m), Cloud Cover, Shortwave Radiation, Wind Speed (10 m), and Wind Direction (10 m).

Additionally, a Clear Sky model was used to obtain hourly global horizontal irradiance at the same locations as the power data. These data points serve the purpose of aiding the models to 'learn' seasonality in training data, according to [7]. A Clear Sky model is a physical model used to estimate irradiance based on a geometric representation of a location on earth and its position relative to the sun. This model receives its name because a main assumption made is that there are no clouds in the sky. There are some extensions to the basic, location-based Clear Sky model that include particle turbidity in the air such as pollution or other aerosols. However, in this case, a basic location-based Clear Sky model was implemented in Python using the popular photovoltaics library *pvlib* [8]. The basic model was implemented because it serves the purpose of helping Machine Learning models to learn seasonality. The output variable that is obtained with this model and used in this project is the global horizontal irradiance (ghi), reported in W/m^2.

Data description and manipulation

Input data, or features, were organized in a dataframe (a common data structure in many programming languages) where null values were eliminated. Only data between the time range 7:00-18:00 were considered, since most of the samples outside this time range are null or zero. The features used for this project are:

Table II. Features

Feature name	**Units**
Power	$W \text{ or } kW$
Temperature	^{o}C
Relative Humidity	%
Cloud Cover	%
Shortwave Radiation	W/m^2
Wind Speed	km/h
Wind Direction	^{o}
Clear Sky Global Horizontal Irradiance (ghi)	W/m^2

The timestep for the dataframes in WA and ACT is 10 minutes, while the timestep for the dataframe from Bogotá is 5 minutes. All hourly data was adjusted to these timesteps, but no interpolation was made; hourly data repeats itself in each 10 or 5-minute timestep that corresponds to each hour. This was done because it maintains the size of the power data's timestep, so all forecasting performed in this project is still considered short-term. If, on the contrary, power data is adjusted to the hourly weather variables, the forecasts would have a 1-hour timestep, and that is undesirable for this study.

The output data are the power values for the next time step corresponding to each feature: 10 and 5 minute-ahead power production for Australian and Bogotá locations, respectively. Output data is represented by 3d plots as shown in figure 3:

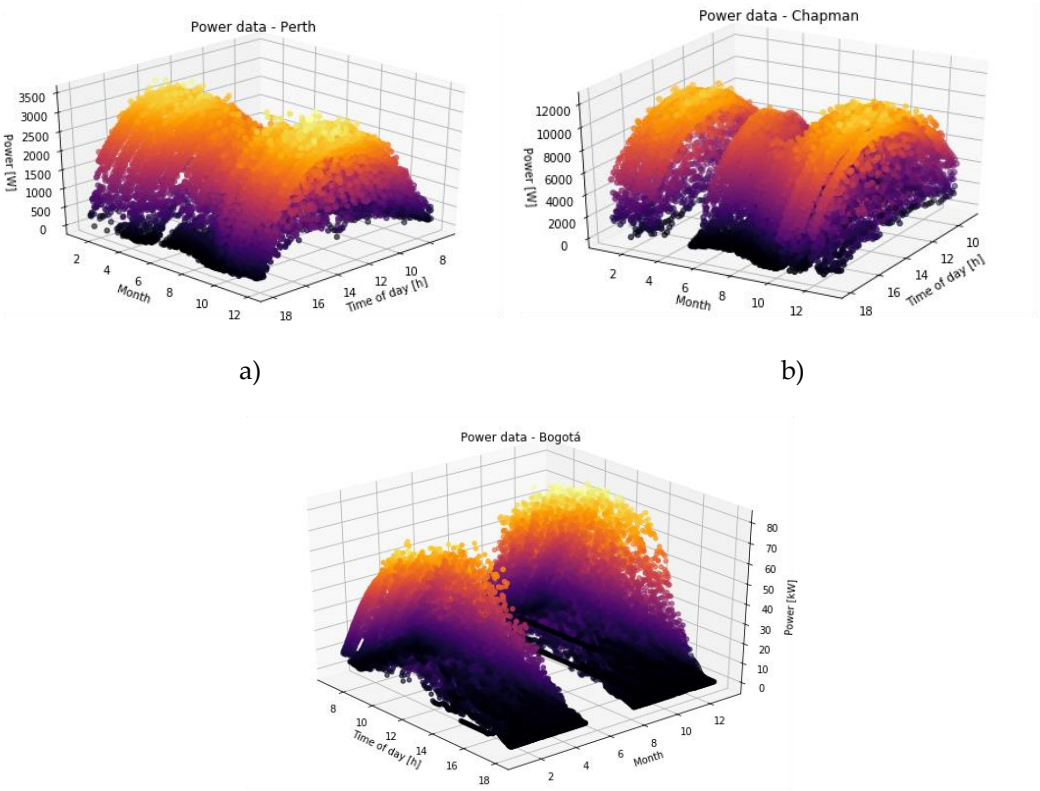
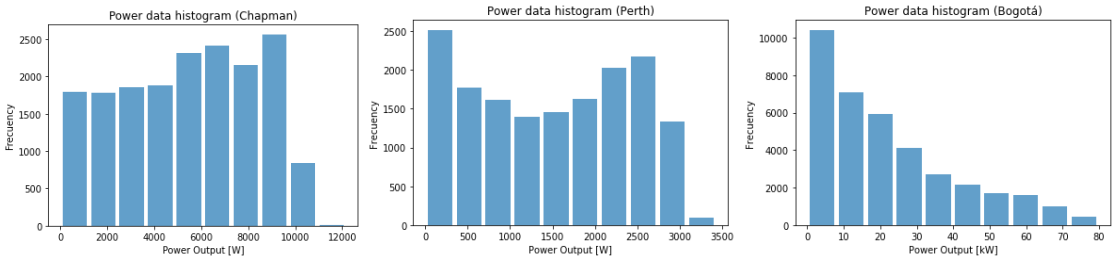


Figure 3. Power data representation in a) Perth, b) Chapman and c) Bogotá.

In order to obtain good results from the Machine Learning models shown in section IV, it is important to ensure output data is balanced. In other words, it should be evenly distributed. This is relevant because, when data is split into training/testing sets, there should be uniform amount of low and high-power values in both sets to avoid biasing either training or testing sets with only high or low values. For example, if a model is trained only with high-power data points, it will learn and therefore only perform well on high-power testing data. The histograms in figure 4 show this graphically, except for Bogotá, which seems to be distributed exponentially.



a) b) c)

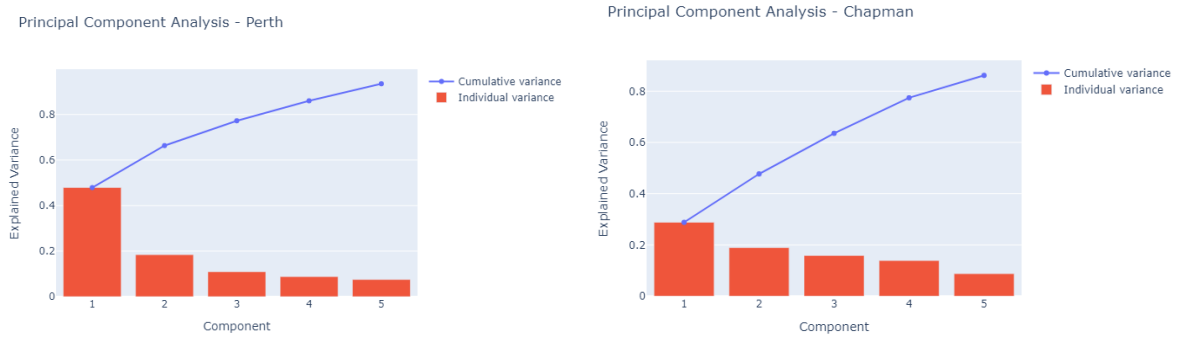
Figure 4. Power data histogram in a) Chapman, b) Perth and c) Bogotá.

III. Feature Selection

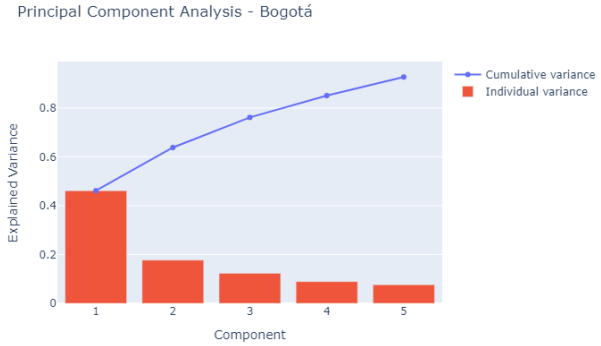
The next step in model building is reducing the dimensionality of the features. This reduction improves Machine Learning model performance in general. This process, known as *dimensionality reduction*, is a type of unsupervised learning because it does not depend on the output data, only the correlation between the original input features [9].

One of the most widely used algorithms is called Principal Component Analysis (PCA) and consists in rotating the datasets such that the correlation between features is minimized. After said rotation, a linear combination of the original features into a lower dimensional set of new features is performed. These new features are called principal components and since they are a linear combination of the original features, the physical interpretation of each individual principal component is often difficult to make. This dimensionality reduction has the goal of maintaining the explained variance of the original dataset while reducing dimensions, which can be understood as eliminating some of the original features. Usually, obtaining an explained variance above 85% from a PCA is acceptable.

After performing a PCA on all three datasets, the resulting explained variance can be observed in the following figures, both component-wise and the cumulative variance using five principal components.



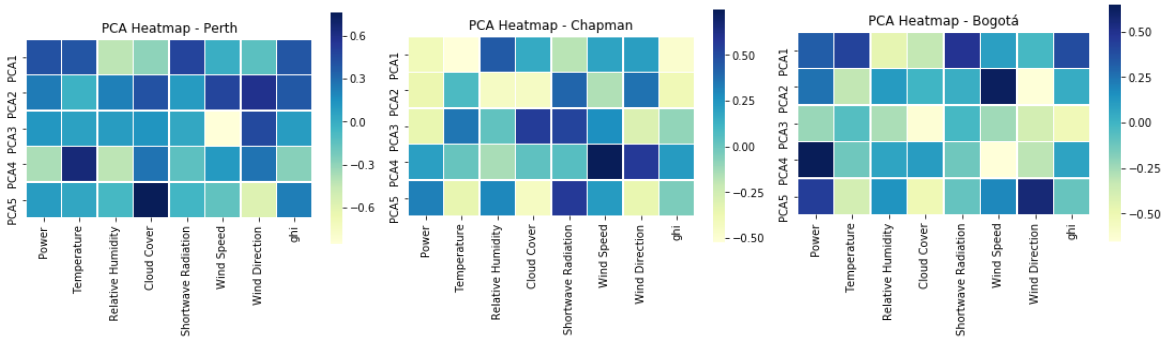
a) b)



c)

Figure 5. PCA Explained variance in a) Perth, b) Chapman and c) Bogotá.

The linear combination of the original features resulting in each principal component is shown in the following heat maps.



a) b) c)

Figure 6. PCA feature heatmap in a) Perth, b) Chapman, and c) Bogotá

PCA results for each location are summarized in table III:

Table III. Cumulative explained variance after performing PCA.

Location	**Total explained variance**
Perth	0.93489
Chapman	0.86156
Bogotá	0.92662

IV. Supervised Learning Models

Supervised Learning has been a widely implemented type of Machine Learning to solve regression problems. The goal of a supervised learning regression model is to predict an output quantity based

on one or more inputs, commonly known as features. In order to build these models, it is necessary to have a set of historical data, where the values of the input features are known as well as the values of the output variable. With enough data samples, models can 'learn' how to make accurate predictions based on input data of which the corresponding output is unknown. This learning stage in the development of a model is usually referred to as 'model training' [8].

It is necessary to have a large amount of data samples because supervised learning models must be trained with them in order to perform well when given unknown data samples and generate accurate predictions. This training process is done in order to minimize the error associated with the prediction the model does on training data samples, commonly known as 'loss'. Once training is done, the model must be tested on data samples that have not been used in training in order to observe how well the model performs on new data. In this sense, datasets are normally split into training and testing data.

In this case, all datasets were split randomly such that 80% of samples correspond to training data and 20% to testing data. Because data is split randomly, an inevitable assumption made when building the models for this project is that each data point is independent from previous samples and predictions, given that randomization breaks the order of the time series.

Model performance is quantified with the coefficient of determination (R^2) in every case. As the value for this performance metric approaches 1, model performance is better. When $R^2 = 1$ there is a perfect fit; an unrealistic situation. Commonly, a model with a determination coefficient above 0.8 is considered a good fit.

In the following sub-sections, a brief description and main results are shown for each model built in this project. A complete mathematical demonstration of these models is not shown in this paper since that is not part of the scope of this project.

Kernelized Support Vector Regression (SVR)

A Support Vector Regression is a supervised learning algorithm based on Support Vector Machines (SVM), one of the most popular Machine Learning algorithms used for classification purposes. A SVR model is also very similar to a linear regression but includes a slack hyperparameter that allows for errors between the model's predictions and the data to fall into an acceptable range [8].

This model is described by the linear equation that relates input data (x') to output values (y) with a parameter β and b.

$$y = \beta x' + b \tag{1}$$

The goal of the model is to minimize β while ensuring that all training samples deviate from the line described by equation 1 by a value less than ε. However, since sometimes this optimization problem is sometimes unfeasible, slack variables ξ and ξ^* are introduced such that some points will be able to fall outside the region defined by +/- ε [10]. This is shown clearly in figure 7.

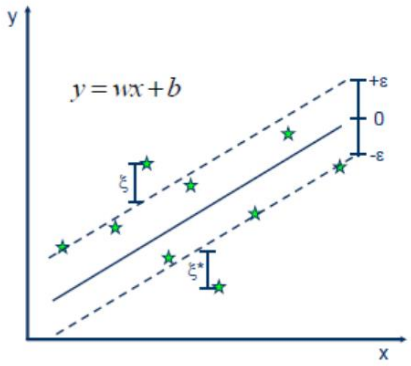


Figure 7. SVR Model. Taken from (https://www.saedsayad.com/support_vector_machine_reg.htm)

Thus, the optimization model is expressed as:

$$\min \frac{1}{2}\beta + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{2}$$

$$subject\ to:$$

$$\begin{gathered} y_n - (\beta x_n' + b) \leq \epsilon + \xi_n\ , \forall\ n \in N \\ (\beta x_n' + b) - y_n \leq \epsilon + \xi_n^*\ , \forall\ n \in N \\ \xi_n^* \geq 0\ \forall\ n \in N \\ \xi_n \geq 0\ \forall\ n \in N \end{gathered} \tag{3}$$

Where C is a constant that determines the penalty given to outliers: when C is large more outliers are tolerated but that may lead to overfitting. N is the set of training samples.

However, in this project SVR data is transformed with a non-linear function in order to properly represent the non-linear nature and complexity of PV power production. This transformation is commonly referred to as the 'kernel trick' and is further explained in [8]. The kernel transformation that seemed to improve model performance the most was a 5th degree polynomial function. Results for this regression are shown in table IV. The model was implemented using scikit-learn.

Table IV. SVR model performance by location.

Location	**Training R^2**	**Testing R^2**
Perth	0.8509	0.8523
Chapman	0.8143	0.7982
Bogotá	0.8357	0.8343

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors algorithm is commonly used for both classification and regression problems because it is mathematically simple to implement and understand. In its simplest form, the KNN algorithm searches for the nearest training sample for any given input features for which a prediction must be made and assigns the training value to the prediction. In this simple case, the value of K (number of neighbors to search for) equals 1. When a larger number of K is considered, the

prediction is assigned a weighted average of the value of the K nearest training samples [8]. So, prediction values (y′) in a given set N, based on the input value x′ can be calculated from the K nearest neighbors using equation 4:

$$y'_n = \frac{1}{K} * \sum_{i=1}^{K} w_i y_i \ , \forall\, n \in N \tag{4}$$

Where w_i represents the weights assigned to each K-neighbor.

To find the nearest neighbor, Euclidian distance is calculated. In a n-dimensional space, if point $a=(a_1,a_2,\ldots,a_n)$ and point $b=(b_1,b_2,\ldots,b_n)$, distance $d_{a\to b}$ is calculated as:

$$d_{a\to b} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{5}$$

In this project, to determine which training sample is closest to the test sample, Euclidian distance is calculated and a simple non-weighted average among the K neighbors assigns the value to the prediction. To determine which value of K yields the best results, an iteration between K=2 to K=10 is performed and plotted. The best value of K is when the test R^2 score is maximized.

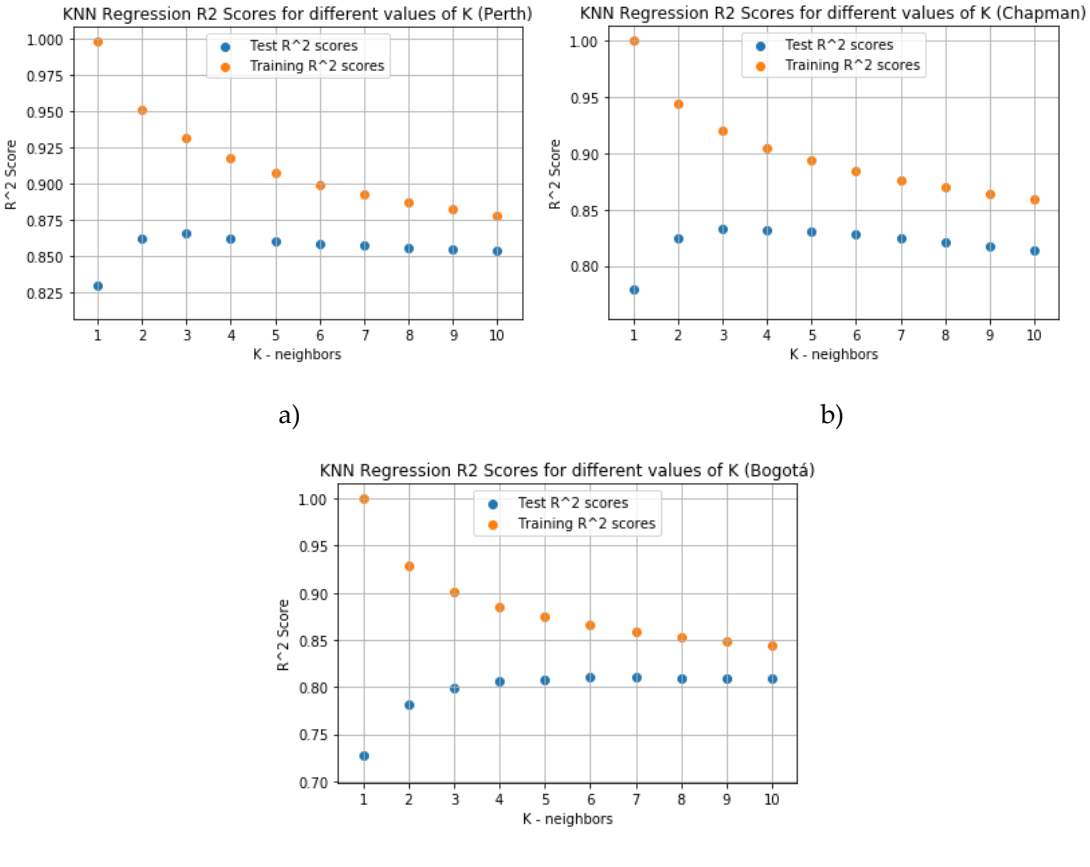


Figure 8. KNN iterations to obtain the optimal value of K in a) Perth, b) Chapman, and c) Bogotá.

Table V. KNN model performance by location.

Location	**Optimal value of K**	**Test R^2 score**
Perth	3	0.8655
Chapman	3	0.8326
Bogotá	6	0.8100

Random Forest (RF)

A Random Forest Regression is an algorithm that is a group of Decision Trees, a simpler algorithm that consists in a series of if/else conditions that lead to a decision or output value. The amount of if/else conditions in the model is usually referred to as depth and determines the model's complexity. However, depth is a problematic hyperparameter that must be properly selected because, on one hand, if depth is shallow (i.e. a small value) the model is not capable of learning complex patterns and therefore it is not a particularly interesting or useful model. On the other hand, if the model is too deep (i.e. the depth is a large value) the model 'memorizes' all the training data, which leads to a perfect performance on training data but poor performance on new data – this problem is known as overfitting. Individual Decision Trees' main setback is their tendency to overfit data easily. A simplified example (with depth=1) is represented in figure X for explanatory purposes. In that example, all data samples with ghi greater than 300 W/m² are assigned to the node in the left, while the rest are assigned to the node in the right. Then, when the model is assigned input data to make a prediction, it verifies if ghi is greater than 300 W/m² or not. Depending on the condition, the prediction is assigned the average power value of all training samples that fulfill the condition based on ghi.

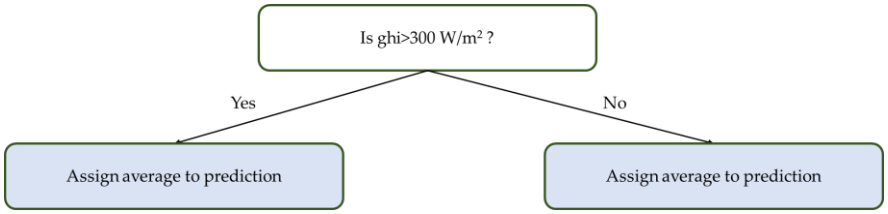


Figure 9. Decision tree example.

When Decision Trees are grouped into a Random Forest, risk of overfitting tends to decrease. This happens because a RF creates many Decision Trees that each perform differently, so when an average of the results of all the different trees is calculated, risk of overfitting decreases [8].

In this project, a Random Forest was built based on tuning two main hyperparameters: Number of trees and tree depth. As the number of trees is increased, model performance should always increase but so does training time. In this case, the number of trees was set to 100 because beyond that number, model performance did not improve significantly. Ideal tree depth was found iterating through different depth values such that overfitting is minimized - the difference between training and testing R^2 must be as small as possible – and model performance is acceptable.

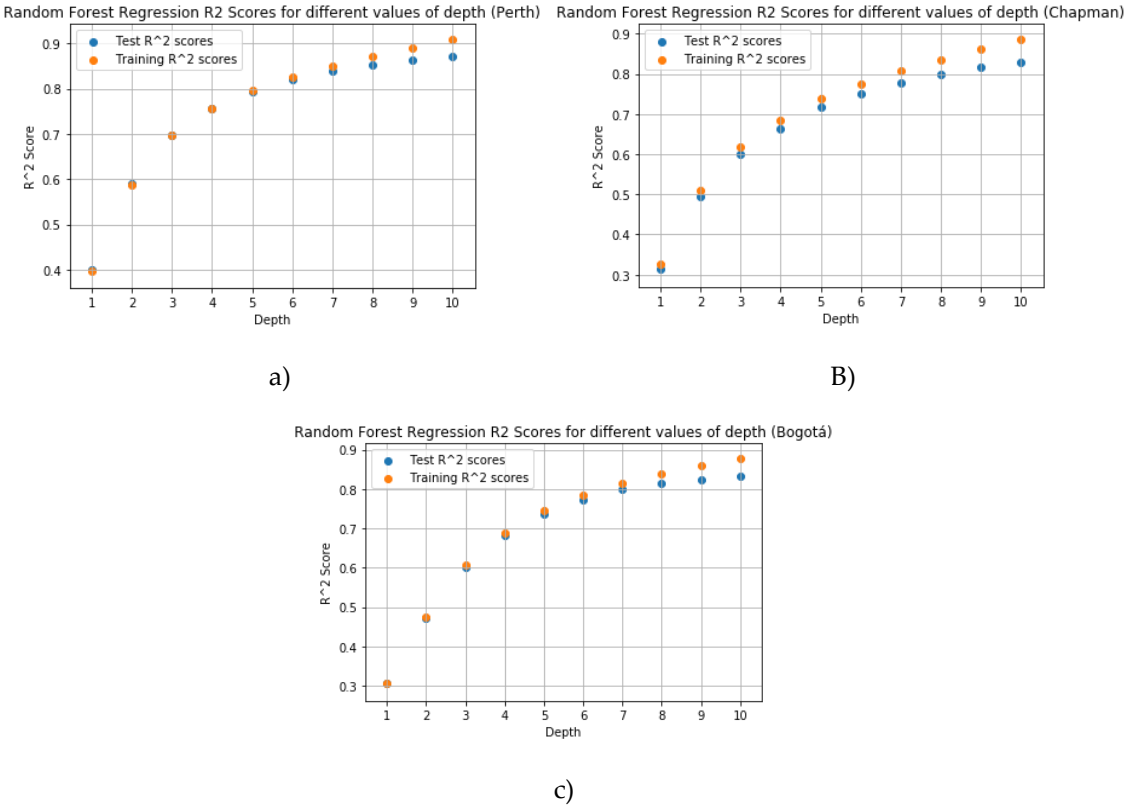


Figure 10. Random Forest depth iteration with 100 trees in a) Perth, b) Chapman, and c) Bogotá.

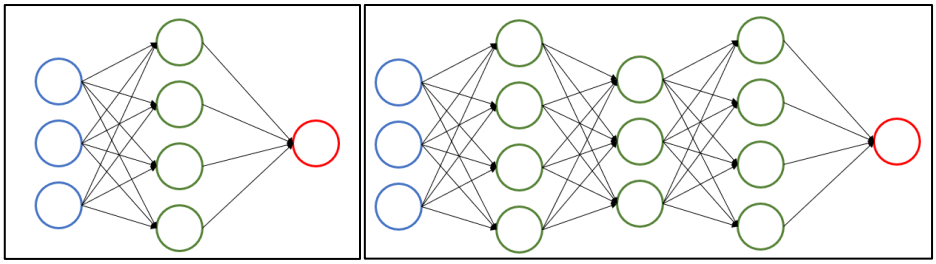
The best value of depth for each location is selected visually and the resulting model performances are shown in table VI.

Table VI. RF model performance by location.

Location	**Tree Depth**	**Training R^2**	**Testing R^2**
Perth	8	0.8706	0.8526
Chapman	6	0.7757	0.7507
Bogotá	7	0.8154	0.7991

Single Layer and Deep Neural Networks (NN)

Neural Networks (NN) are one of the most popular type of Machine Learning models today because of their excellent performance and infinite different variations, which lead to a high versatility in both classification and regression problems. Basic Neural Networks have one or more hidden layers stacked in front of the first input layer, as shown in figure 11. A Single Layer NN is a network with only one hidden layer (figure 11a) , while a Deep NN is a network with many stacked hidden layers (figure 11b). Deep NN have become increasingly popular and are the basic forms of a new type of Machine Learning algorithms known as Deep Learning. Naturally, Deep NN are more complex than Single Layer NN which leads to better performance on more complex data. Also, Deep Learning models tend to perform better when the amount of training sample increases, hence the growing interest on what is known as 'Big Data', which is simply datasets with large amounts of samples – usually in the hundreds of thousands, millions, or more [9].



a) b)

Figure 11. Neural Network representations for a) Single Layer NN, and b) Deep NN. Blue circles represent the input features, green circles represent hidden neurons, and the red circles represent the output variable. The connections between circles represent weights.

Forward propagation is performed through the layers to calculate the output variable for each training sample. After that, weights are adjusted in order to minimize loss through a process called back propagation. This forward and back propagation process is iterated through all the training samples. NN architectures can be defined very briefly by the number of hidden layers (L), the number of neurons in each hidden layer (N), and the activation function in each layer (a).

Forward propagation consists in performing calculations on data through each node or neuron in the network, from left to right. In general, each neuron in each layer performs two calculations based on the following parameters: the previous layer neuron's output ($a_n^{[L-1]}$), the corresponding weight between any two neurons ($w_n^{[L]}$), an activation function ($a_n^{[L]}$), and a bias constant (b). The first calculation yields the value of $z_n^{[L]}$, shown in equation 6. [11]

Note: the superscript surrounded by square brackets denotes the layer a neuron is in and the subscript denotes the position of a neuron within a layer.

$$z_n^{[l]} = w_n^{[l]} a_n^{[l-1]} + b \; \forall \; l \in L, n \in N \tag{6}$$

The second calculation performed by a neuron is the activation function, which is a hyperparameter denoted by g(z). Therefore, the output of any neuron is its activation function. In this case, a rectified linear unit (ReLU) activation function is used:

$$a_n^{[L]} = g^{[l]}(z_n) \tag{7}$$

$$g^{[l]}(z_n) = \max\,(0, z_n^{[L]}) \tag{8}$$

Forward propagation is performed across neurons until reaching the output neuron. The result of the output neuron's activation function is the prediction made by the neural network.

To determine if a prediction performed by the neural network ($\hat{y}_i$) is close to the actual value (y_i), a loss function (L) is defined. There are many different types of loss functions, but one of the most popular is the mean squared error (MSE), described by equation 9, where N represents the set of training samples.

$$L(y_i, \hat{y}_i) = MSE = \frac{1}{N} * \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{9}$$

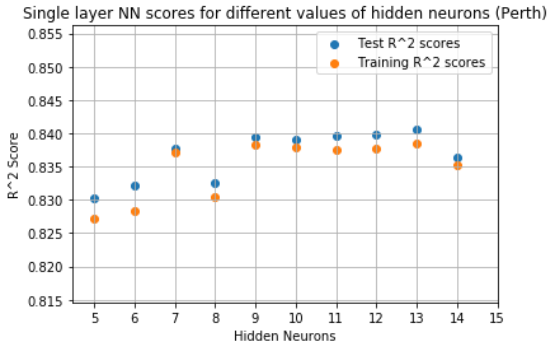
A process called gradient descent is performed to update the parameters of the neural network. The calculations, known as backpropagation, are performed from right to left across the network for each training sample. The parameters that must be updated are the weights (w) and the bias constants (b) and are grouped in a single variable θ(w,b). Their impact on the loss function, represented by J(θ), is calculated using derivatives. This impact is referred to as the gradient of the loss function (∇J). Neural Networks usually have thousands of parameters that must be calculated, so all calculations are made using matrices. The explanation made in this section is a simplified version of what is found in [11].

$$\nabla J = \begin{cases} \frac{\delta J}{\delta w} = \frac{\delta J}{\delta a}\frac{\delta a}{\delta z}\frac{\delta z}{\delta w} \\ \frac{\delta J}{\delta b} = \frac{\delta J}{\delta a}\frac{\delta a}{\delta z}\frac{\delta z}{\delta b} \end{cases} \tag{10}$$

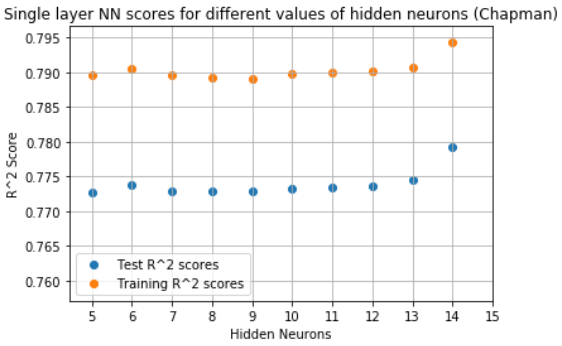
Parameters are updated across each training sample based on the hyperparameter of learning rate α, which in this case was 0.0001.

$$\theta := \theta - \alpha * \nabla J \tag{11}$$

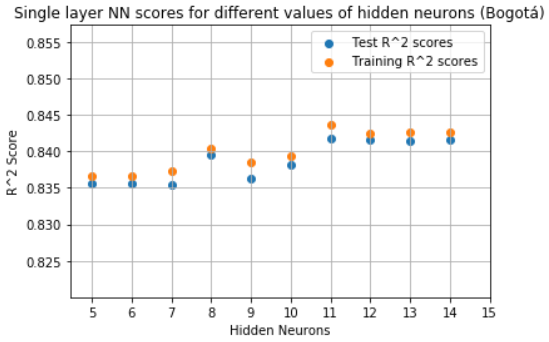
In this case, a procedure to find an ideal number of hidden neurons for the Single Layer NN was implemented. Increasing the number of neurons in a layer tends to improve performance, but after reaching a threshold, when there are too many neurons, training time and model complexity increase while gains in performance do not. Hence, having too many hidden neurons in a single layer is unjustifiable. All NN in this project were built with the Rectified Linear Unit (ReLU) activation function.



a)



b)



c)

Figure 12. Single Layer NN performance scores for different numbers of hidden neurons in a) Perth, b) Chapman, and c) Bogotá.

The selected number of neurons for each location is obtained graphically and results are shown in the following table:

Table VII. Single Layer NN model performance by location

Location	**Hidden Neurons**	**Training R^2**	**Testing R^2**
Perth	7	0.8371	0.8377
Chapman	14	0.7943	0.7791
Bogotá	12	0.8424	0.8415

A Deep NN is then constructed with the first three layers having the same number of hidden neurons as the Single Layer model shown previously. The next three layers have more hidden neurons because according to [9], deep layers can calculate more complex functions and therefore should have more neurons as the first few layers.

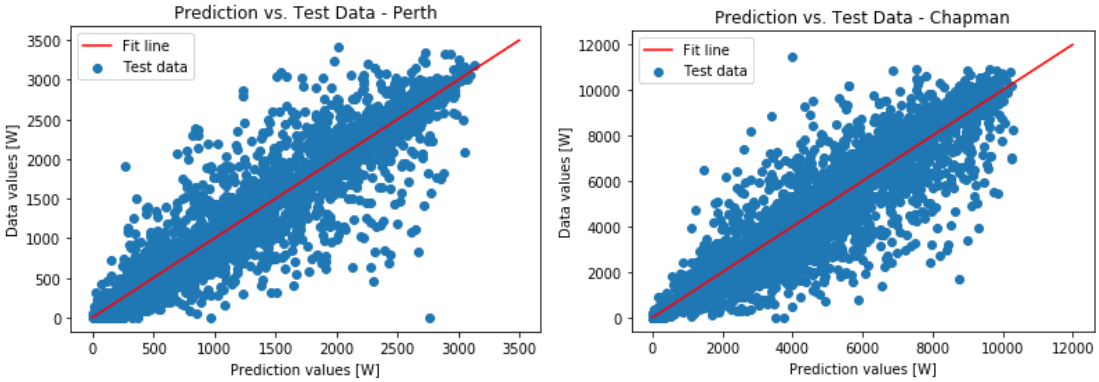
The results for a 6 Layer-Deep NN are shown in table VIII.

Table VIII. Deep NN model performance by location.

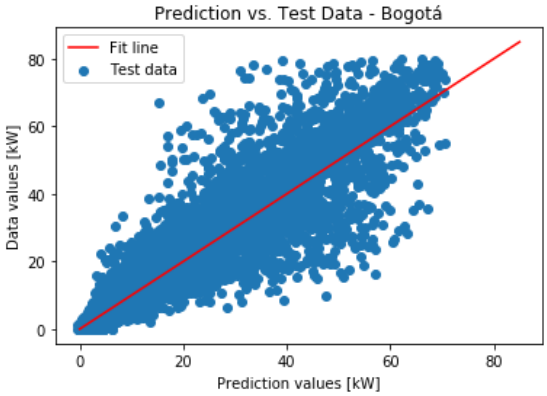
Location	**Training R^2**	**Testing R^2**
Perth	0.8419	0.8419
Chapman	0.8420	0.8271
Bogotá	0.8460	0.8421

V. Results and Model Comparison

In order to visualize how predictions behave, two types of plots are presented based on the results of the best model for each location: A KNN in Perth and Deep NN for Chapman and Bogotá. First, a fit plot is shown, where vertical and horizontal axes represent the real data values and the predictions, respectively. Ideally, points in this plot should fall near to a line with positive slope of 1 and samples on the line represent a perfect fit.



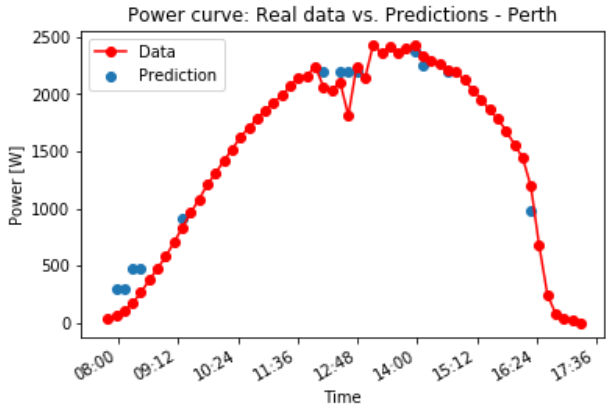
a) b)



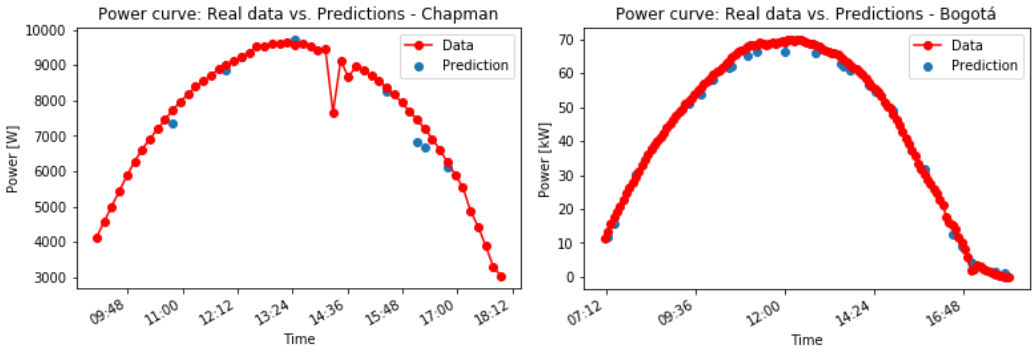
c)

Figure 13. Fit of prediction vs. test data in a) Perth, b) Chapman, and c) Bogotá.

The second type of plot requires reordering testing and training data that was previously split randomly. This plot shows the power production curve vs. time of any day across the complete dataset. The plot also shows predictions for testing samples in any given day. Different days were selected for visualization in each location.



a)



b) c)

Figure 14. Power curve showing predictions vs. real data in a) Perth, b) Chapman, and c) Bogotá.

Finally, a comparison between all models across all locations is made:

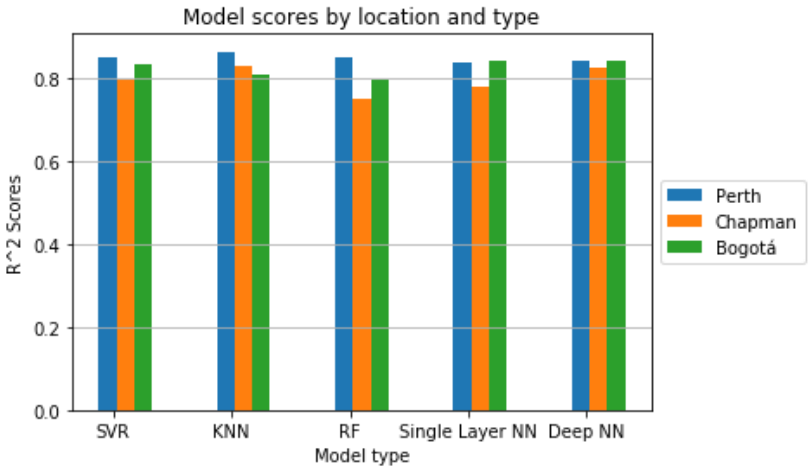


Figure 15. Model comparison by type and location.

VI. Discussion of results

Results show that Machine Learning algorithms built in this project do not differ greatly from one another, considering all the location datasets that were used, as seen in Figure 13. It is also clear that all models except for the Single Layer NN performed best in Perth. The reason for this might be that data from Perth seems graphically to be the most complete across the entire timeseries – there is no month in which there is a clear 'hole' of missing data, as seen in Figure 3a. On the other hand, models that were fed with data from Chapman seem to perform the worst, data which has a clear hole representing missing data in figure 3b. Overall, Deep NN seem to be the best model because performance results are all above 0.8 and seem to be similar for each location. KNN models also perform well, but there is a clear difference in performance between Perth and Bogotá. Figure 12 shows that models are properly learning patterns for PV power because predictions (blue points) fall very close to the real data power curve.

Figure 13 shows a positive tendency between real data and predictions – predictions seem to be well fitted to the real data. This is clear because most scatter points are close to the line with slope 1. This means that the models are good making predictions, despite some outliers (data points far from the red line).

It was mentioned in this paper that ensuring data is evenly distributed is important for training. However, results from Bogotá - in which data was exponentially distributed – showed good performance compared to the other datasets. This may be happening because the training-testing split was performed randomly, so enough data samples associated with high, medium, and low power production are used for training. Therefore, models seem to properly learn patterns in scenarios where power production is low, medium, and high.

The study available in [4] obtains similar results as this study in terms of the determination coefficient and Neural Networks, except the predictions they make have a 24-hour temporal resolution. They obtain an average R^2 of 0.81 on a 1-3-year experiment using NARX Neural Networks. However, as the experiment duration increases beyond 3 years, their model performance increases to 0.867 using a 10-year long dataset. Also, that study implements a Bayesian hierarchical modeling using an extended Kalman filter (Bayesian-EKF) which yields an average R^2 of 0.959. Considering the level of complexity of the models developed in this study, our results are satisfactory.

VII. Conclusions and future work

As discussed in section VI of this paper, having better datasets can lead to better performance. A PV power dataset can be considered as 'good' if it has a large amount of data samples with a small timestep in order to make short term predictions, data samples are continuous in time (i.e. no missing values), and data is properly preprocessed before training supervised learning models. Many Machine Learning researchers say data manipulation is the most important part of the methodology used to solve problems using Machine Learning.

Machine Learning models trained in this project were capable of generating accurate short-term predictions only on roughly one year of PV data. This shows that Machine Learning is a potentially powerful tool to generate predictions for the renewable energy industry, since performance is expected to improve even more with more data samples. This can be a promising way to improve reliability of renewable energy systems, one of their main flaws in the past.

Also, an attractive characteristic of these models is their relative ease to implement, given data preprocessing is properly done. In this project, models were implemented with Python library Sci-kit Learn, a user-friendly Machine Learning library. Simplicity is also achieved because this methodology allows the researcher to skip certain technical steps used for a traditional PV system analysis such as panel inclination, inverter configuration, among others.

As with many other Machine Learning applications, Deep Learning seems to be the most consistent ways to solve problems, especially because Deep Learning algorithms perform even better on large amounts of data. In this sense, future work in this field should be oriented towards developing more complex Deep Learning models that are specifically designed for timeseries analysis, such as Long Short-Term Memory (LSTM) Neural Networks.

References

[1] IRENA, «Renewable Power Generation Costs 2018,» Abu Dhabi, 2019.

[2] Vox News, «The 'duck curve' is solar energy's greatest challeng,» May 2018. [Online]. Available: https://www.youtube.com/watch?v=YYLzss58CLs [. [Last access: March 2019].

[3] K. Jäger, O. Isabella, A. H. Smets, R. A. van Swaaij and M. Zeman, Solar Energy Fundamentals, Technology, and Systems, Delft: Delft University of Technology, 2014.

[4] S. Hussain and A. AlAlili, «Online Sequential Learning of Neural Networks in Solar Radiation Modeling Using Hybrid Bayesian Hierarchical Approach,» *Journal of Solar Energy Engineering,* vol. 138, 2016.

[5] W. Lee, K. Kim, J. Park, J. Kim and Y. Kim, «Forecasting Solar Power Using Long-Short Term Memory and Convolutional Neural Networks,» *IEEE Access,* vol. 6, 2018.

[6] J. M. Bright, S. Killinger and N. A. Engerer, "Data article: Distributed PV power data for three cities in Australia," *Journal of Renewable and Sustainable Energy,* no. 11, 2019.

[7] C. Feng and J. Zhang, «Hourly-Similarity Based Solar Forecasting Using Multi-Model Machine Learning Blending,» IEEE, Portland, 2018.

[8] W. F. Holmgren, C. W. Hansen and . M. A. Mikofski, «pvlib python: a python package for modeling solar energy systems.,» *Journal of Open Source Software,* vol. 884, nº 29, p. 3, 2018.

[9] A. C. Müller and S. Guido, Introduction to Machine Learning with Python, Boston: O'Reilly Media, 2016.

[10] MathWorks, inc, «Understanding Support Vector Machine Regression,» [Online]. Available: https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html#:~:text=Linear%20SVM%20Regression%3A%20Primal%20Formula,-Suppose%20we%20have&text=subject%20to%20all%20residuals%20having,these%20constraints%20for%20all%20points.. [Last access: June 2020].

[11] A. Ng, «Neural Networks and Deep Learning,» deeplearning.ai, Palo Alto, 2020.

[12] R. Bayindir, . M. Yesilbudak, M. Colak and N. Genc, «A Novel Application of Naïve Bayes Classifier in Photovoltaic Energy Prediction,» IEEE, Cancún, 2017.

[13] K. Chen, Z. He, K. Chen, J. Hu and J. He, «Solar Energy Forecasting With Numerical Weather Predictions on a Grid and Convolutional Neural Networks,» from *IEEE Conference on Energy Internet and Energy System Integration*, Beijing, 2019.

[14] C. Chen, D. Shanxu, C. Tao and L. Bangyin, «Online 24-h solar power forecasting based on weather type,» *Science Direct,* 2011.

[15] A. Laouafi, M. Mordjaoui and D. Dib, «One-Hour Ahead Electric Load and Wind-Solar Power Generation Forecasting using Artificial Neural Network,» de *6th International Renewable Energy Congress (IREC)*, Sousse, 2015.

[16] D. Lee and K. Kim, «Recurrent Neural Network-Based Hourly Prediction of Photovoltaic Power Output Using Meteorological Information,» *Energies,* 2019.