Evaluation of Machine Learning Methods on Large-Scale Spatio-Temporal Data for Photovoltaic Power Prediction

by
Evan E. Sauter

Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE

A thesis in partial fulfillment for the
Degree of Master of Science
in
Electrical and Computer Engineering

April 2023

APPROVED:

Dr. Ziming Zhang:

Dr. Gregory Noetscher:

Dr. Rodica Neamtu:

Abstract

The exponential increase in photovoltaic (PV) arrays installed globally, particularly given the intermittent nature of PV generation, has emphasized the need to accurately forecast the predicted output power of the arrays. Regardless of the length of the forecasts, the modeling of PV arrays is made difficult from their dependence on weather. Typically, the model projections are generated from datasets at one location across a couple of years. The purpose of this study was to compare the effectiveness of regression models in very short-term deterministic forecasts for spatio-temporal projections. The compiled dataset is unique given it consists of weather and output power data of PVs located at five cities spanning three and six years in length. Grated recurrent unit (GRU) generalized the best for same-city and cross-city predictions, while long short-term memory (LSTM) and ensemble bagging had the best cross-city and same-city predictions, respectively. The code and data are available at https://github.com/Zhang-VISLab?tab=repositories.

Acknowledgements

I am grateful to Dr. Ziming Zhang, Dr. Gregory Noetscher, Dr. Rodica Neamtu, Dr. Maqsood Mughal, and Mr. Yiqing Zhang from Worcester Polytechnic Institute for the insight and guidance that they provided.

Contents

List of Figures

List of Tables

Acronyms

ANN artificial neural network. 8, 9

CNN convolutional neural network. 10

DHI diffused horizontal irradiance. 7, 13, 26

DNI diffused normal irradiance. 7, 13, 25, 26

GHI global horizontal irradiance. 7, 9, 13, 25, 26

GRU grated recurrent unit. 1, 10, 19, 21–23, 28, 34, 36

KNN k-nearest neighbors. 10, 14

LSGDR linear stochastic gradient descent regression. 10

LSTM long short-term memory. 1, 10, 19–23, 34, 36

MLP multi-layer perceptron. 10, 19, 20

MSE mean squared error. 22–24, 34, 35

NREL National Renewable Energy Laboratory. 11, 13

PLS partial least squares. 10, 15, 16

PV photovoltaic. 1, 5–9, 11, 12, 14, 17, 24–26, 36

RES renewable energy sources. 5–7, 36

RMSE root mean squared error. 3, 24, 26, 28, 32–35

RNN recurrent neural network. 10, 19

SIG Energy Special Interest Groups Energy. 11, 12

SVM support vector machine. 10

SVR support vector regressor. 10, 17

WPI Worcester Polytechnic Institute. 1, 11

1 Introduction

1.1 Problem

The usage of renewable energy sources (RES) within the energy sector has been exponentially increasing, both globally and domestically, within the twenty-first century. While there are several types of RES, PV arrays have seen consistent improvements in their efficiency and reductions in cost, subsequently leading to becoming more readily adopted. In 2001, only 1.5GW of PV generation were constructed, while the global gigawatt capacity constructed in 2011 and 2021 was 65GW and approximately 156.1GW, respectively [2], [3].

Although the shift from fossil fuels to RES likely originated from the desired to decarbonize our world, the shift would not have gained much traction without the reduction in costs for these alternative sources. There are multiple factors that govern the cost of a given PV system. The most important economic inputs are the cost of the system and the energy that it would deliver each year [4]. These factors are closely followed by the energy displaced by the system, if there are any tax credits or other economic incentives, and how the system will be paid for. Additional considerations include the cost for operation and maintenance, the future cost of electricity from utilities, any loan terms or if the system was purchased outright, the lifetime of the system, and the cost for removing the system after the lifespan is achieved [4]. The estimated total selling price per peak DC watt of power generated by PV arrays across the residential, commercial, and utility sectors were projected to significantly decrease from 2010 to 2020 [4]. The prices were \$5.71, \$4.59, \$3.80 in 2010 for residential, commercial, and utility and were forecasted to drop to \$1.50, \$1.25, \$1.00 in 2020, respectively.

With this trend in the reduction of the overall cost of PV arrays across all sectors, the rate of installation and construction of additional arrays would naturally increase. The projected growth of RES from 2020 to 2026 is expected to increase by more than 60 percent such that the total global generation is more than 4,800GW [3]. To put this in another perspective, this would amount to more than the current global generation of fossil fuels and nuclear power combined. However, unlike fossil fuels and nuclear power generation, RES are subject to variability and inconsistencies due to weather factors [2]. Therefore, there is value in applying machine learning algorithms to model PV arrays, such that grid operators can accurately predict the output power at a given instance to ensure the load demand is met.

1.2 Significance

Given the inconsistencies inherent to RES due to their reliance on weather and external factors, a diverse collection of RES and predictive modeling of these respective sources is necessary to maintain the load demand of electric grids. While all RES must receive careful consideration in terms of potential installation locations, for PVs, the impact of partial shading and typical weather patterns of the region are critically important [5]. Partial shading is impactful to the output power of an array because the cells that compose each panel are particularly sensitive to shading. If a cell is minimally shaded, neighboring cells can be affected as well because most arrays are composed of strings of cells [4]. Therefore, even if a small amount of shading can significantly compromise the output power. The standard procedure to mitigate this behavior is through the implementation of bypass diodes. If one cell in a string cells are shaded, rather than simply not contributing to the net output power of the array, the shaded cell can produce a negative voltage [4]. When this occurs, it is best to remove the shaded cell from the resulting circuit rather than to drive current through the whole string, thereby impairing the net power output. For this reason, and to prevent system failure, while also maintaining load demands by mitigating the impacts of shading are a few of the reasons it is critical to carefully construct these systems.

Therefore, PV generation is most closely tied to cloud cover. When there are no clouds, PV generation follows a diurnal curve as the sun traverses the sky, which is both smooth and predictable [6]. When clouds are present, they impact both the quality and quantity of the output power generated. One such example is when there are sparse cumulus clouds in an otherwise relatively clear day. Due to the sparsity of these clouds, the shading upon the given PV array is inconsistent [6]. The quality of the output power is then greatly diminished while the quantity remains relatively high. This contrasts with when there are opaque stratus clouds that linger for hours, thereby causing the output power to be greatly diminished while the quality remains high [7]. Whether a cloud is drifting such that it is just shading an array or that it is just departing, the abrupt nature of this change creates a step change that either is a decrease or increase of the power generation [8]. This is classified as ramping and is a factor that grid operators must consider when balancing load demands.

Any significant surplus or deficit of PV power generation, if multiple sources are tied together, must be balanced with an equal but opposite allocation such that the load demand of a given region is met [6]. The power quality from a PV array can be most easily visualized on the consumer end when voltage flicker occurs. While low power quality is not directly indicative of voltage flicker occurring, it would occur more frequently given increased variations in the quality of power delivered. Voltage flicker is when the demand for electricity momentarily rises above the threshold of power generation that can be delivered [9]. This can most easily be observed in the brief flickering of lights after a large appliance was activated. That said, preventing voltage flicker and moderating ramping effects are just a few of many parameters that govern the output power generated by a singular or collection of PV arrays. These are just a few tasks that are categorized as electric grid management, as shown in Fig. 1. Therefore, the more accurate the projections of output power from RES, the more efficient and cost-effective energy management can be. The grid operators would be able to either pull from energy storage units to meet the grid's load demand or allocate any surplus of energy generated to these battery units. Additionally, with a better understanding of the amount of energy generated by a source at any given point, the grid operator would be able to better transmit the energy to regions with greater demand.

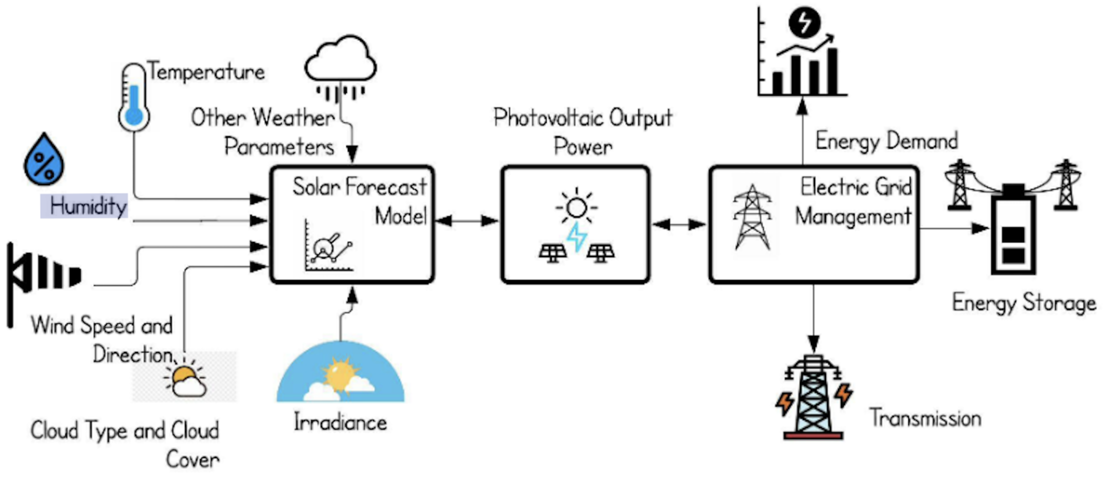


Figure 1: The connection between inputting weather data into a model, to utilizing that model in assisting in the management of the electrical grid. Source [1].

The impact of clouds upon PVs influence the amount of solar irradiance that is received by the arrays. This irradiance can be subcategorized based on the angle and method that it hits each panel of the PV array. These distinctions include the diffused horizontal irradiance (DHI), diffused normal irradiance (DNI), global horizontal irradiance (GHI) and the corresponding clear sky variations. Although the different distinctions of irradiance are grouped as one in Fig. 1, it also becomes evident that there are multiple other parameters that influence the output power of PVs. These include but are not limited to the cloud type and amount of cloud cover, wind speed, relative humidity, temperature, and other weather parameters. Therefore, given these characteristics and the additional complexity of the conditions governing the power generation of PVs, logically these arrays would greatly benefit from the usage of predictive modeling. Whereby, the output power can be modeled given the weather conditions at the location. These forecasts are best accomplished using machine and deep learning.

Artificial intelligence is the parent category that contains machine learning, and within it exists a subset called deep learning. Machine learning can be considered a shallow neural network, as it allows programs to learn from data without being declaratively programmed [10]. Deep learning differs by having several hidden layers, complex connectivity architectures, and different transfer operations [10]. Deep learning algorithms are synonymous with neural networks. The term machine learning was first used in 1959 by the American scientist Arthur Lee Samuel to describe the field of computers that enabled the computers to learn from data without being directly programmed [10]. However, in the contemporary era, machine learning has become universal. Its usage has entered the everyday workspace, household devices, automobiles, and any sphere with the intention of providing ease-of-life to people. From the ubiquitous growth of machine learning, it has also been employed to generate predictive models for the output power of PVs.

1.3 PV Prediction in Literature

There are numerous publications that compare different machine learning and deep learning models against one another. In most cases, there are only a handful of models whose performances are compared, while there are a few publications that are more comprehensive, comparing the results across articles.

However, before determining which model is best to use, one must first understand the different durations of forecasts and their corresponding applications. Forecasts are typically categorized into four groups based on the length of the projection: very short-, short-, medium-, and long-term forecasts [11]. Intraday forecasts are classified as very short-term, and onsite measurements are typically required for these projections. These very short-term forecasts are applied in real-time operations such as spot markets, power smoothening, real-time power dispatching, and automatic generation control [11]. Short-term forecasts are typically between one hour to one week ahead. These projections are typically utilized for reserve optimization, economic dispatching, transmission scheduling, unit commitment, storage system management, and day-ahead markets [11]. Medium-term forecasts typically range from one month to one year [2], while long-term forecasts tend to range from one year to ten years [11]. These longer forecasts are mainly applied to the power sector for determining scheduling and planning within the sector.

To properly compare the accuracy of different models, error analyses and accuracy evaluations were conducted. Under different sky conditions, the accuracy of a model, let alone the baseline, will vary. The California Renewable Energy Collaborative demonstrated that a %RMSE value up to six percent could be expected when forecasting a day ahead under clear sky conditions [12]. While under non-clear sky conditions, %RMSE values of at least 20 percent with a few outliers ranging between 40-80 percent were observed [12]. A study that focused on a type of an artificial neural network (ANN) determined that this type of algorithm could achieve %RMSE values within the range of 15.2-16.3 percent for the day ahead forecasts [13]. Additionally, the forecasting techniques of ANNs on average have been proven to be considerably effective given the inherent ability to record non-linear abrupt changes that are caused by rapid changes in the environmental conditions of the relationship between the input and outputs [11] [13].

1.4 Prior Datasets

As relevant as the forecasting window is to understand the differences between different applications of machine learning upon PV generation, the parameters used also have different predictive capabilities. The parameters of GHI, wind speed forecasts, and ambient temperature are considered basic because they are most often used [14]. That said, the GHI from the actual, hourly, and daily mean GHI, ambient temperature shifted by one hour, azimuth, declination angles, and elevation are the most effective at predictive capability. However, these are considered complex as these are calculated from the basic inputs without requiring additional data [14]. While the daily mean GHI, supplemented by declination angle, azimuth, and modeled 15-min elevation, are useful if operating at a low budget [14]. They come at the cost of a reduced resolution of generated forecasts. In short, the most used predictors for the output power are the average GHI, temperature, wind speed and direction, precipitation, humidity, and cloud cover.

Across two comparative analyses spanning 31 studies, there are significant variations in the sampling rates, lengths of datasets, and locations tested within each study. The average sampling rate was split equally between 15- and 30-minute intervals. There were sampling rates as frequent as once per minute and as infrequent as once an hour [10]. The length of each dataset varied from less than a year to the rare few with five years of data. However, the average length of data available was between one and two years, erring on two years [14]. Furthermore, all but two of the 31 studies had datasets that were restricted to one location. Of the two multi-location studies, they contained data from two and ten different locations.

1.5 Classic Machine Learning Methods

Proportionally, there are more types of machine learning algorithms applied to PV projections than those of deep learning. Typically, the machine learning algorithms of linear regression, ridge, Lasso, elastic net, decision trees, random forests, random forest based ensemble bagging, and support vector machines are applied [10]. In one particular study, an ensemble method that was developed was composed of the elastic net, gradient boosting, and random forest algorithms [10].

1.6 Deep Learning Methods

Deep learning algorithms are tested as frequently as machine learning algorithms. These algorithms consist of multiple layers, each layer composed of multiple nodes, thereby earning the name neural networks. These layers are the input layer, the hidden layer, and the output layer. The hidden layer can be a single layer or multiple layers. Additionally, the output layer may consist of a single node or a multitude of nodes, depending on how many dependent variables are desired. The most overarching type of neural network is the recurrent neural network (RNN). The standard deep learning models within existing studies include multi-layer perceptron (MLP), LSTM, GRU, or a convolutional neural network (CNN) [15]. LSTM models and other RNNs have also been used for these predictions because of their strength when applied to time series datasets [15].

Regardless of the type of the neural network used, the algorithms typically outperform machine learning models. In one study, a joint Siamese CNN and LSTM model (SCNN-LSTM) was developed [15]. This model was then tested against a MLP, a LSTM, and a 3D-CNN. The SCNN-LSTM outperformed the 3D-CNN, which outperformed the LSTM and MLP respectively for 10-minute ahead forecasts. The better performance of the SCNN-LSTM and the 3D-CNN could be attributed to the ability of deep learning models to handle images as well as numerical data. In another study, a Bayesian neural network was tested against a support vector regressor (SVR) and a regression tree [11]. These models were tested under conditions where minimal input features were provided to determine the day-ahead forecasts. Within this study, the three models were trained and tested on the same dataset for an optimal comparitive analysis. The results were that the Bayesian neural network significantly outperformed the SVR which moderately outperformed the regression tree. Another study compared 24 machine learning models for deterministic day-ahead forecasts using a two-year-long dataset at 15-minute resolution [11]. From this comprehensive comparison, it was found that a neural network composed of a MLP had the best overall performance, followed by SVMs.

1.7 Contributions

This study compares the accuracy of projections from 15 algorithms upon expansive spatio-temporal datasets. Of the algorithms, 12 are machine learning, while three are deep learning. The machine learning models analyzed are k-nearest neighbors (KNN), linear regression, linear stochastic gradient descent regression (LSGDR), elastic net, partial least squares (PLS), ridge, kernel ridge, SVR, NuSVR, decision tree, random forest, and ensemble bagging. The neural networks used were MLP, LSTM and GRU. The datasets used in this study included five locations, where all but one of the locations had six years of data. The accuracy of the projections were compared for both same-city and cross-city.

2 Spatio-Temporal Weather and PV Power Data

In this section, the compiled weather and PV dataset is introduced and described. The weather data was obtained from the National Renewable Energy Laboratory (NREL) [16], while the output power of the PV arrays was obtained from the Special Interest Groups Energy (SIG Energy) of California and the University of Massachusetts Amherst [17]. For each of these sources, the usage rights are only restrictive against commercial usage.

2.1 Dataset Analysis

The empirical data for this study was compiled from three different databases. The dataset consists of the weather data and the output power of a given PV array, taken at uniform intervals of date and time. To maintain standardization across the dimensions of the dataset, the weather data from each of the five locations were obtained from NREL [18]. The weather data from NREL is available from the years 1998 to 2021, and it is sampled at a 30-minute refresh rate with a localized region of 4km [16]. The initial city was Amherst, MA, wherein the power data was obtained from the 155kW capacity PV array atop the Computer Science building at the University of Massachusetts Amherst [19]. The locations of the other PV arrays were in the Californian cities of Davis, Huron, Santa Barbara, and La Jolla with output power capacities of 143.2kW, 53.8kW, 42.5kW, and 41.7kW, respectively [20].

Given the vast distances between some of the chosen cities, it is necessary to provide the rationale behind the selection. The city of Amherst, MA, was initially selected to serve as a ground truth for the weather and PV power data collected by an undergraduate capstone project at WPI given the relative proximity of the cities. Wherein this capstone, an array of irradiance sensors were utilized to predict the output power of a small PV array. Given the initial plan had been to train the model on the data from Amherst and to test it on the data collected by the irradiance sensors, a sampling window from 10am to 3pm with a 30-minute refresh rate had been proposed. As this was deemed an acceptable window given the setup and management of the undergraduates. The direction of this research changed, and with it the number and the scope of the cities included. This evolution was done to enable cross-city projections, where a given algorithm was trained on data from one city and tested on another. In order to avoid the persistence of weather in a localized region, it was necessary to locate additional PV arrays outside of New England with a comparable sampling frequency and length of data available. The cities in California were selected based on these criteria. Additionally, the capacities of the PV arrays were large enough that comparisons could be made. The city of Davis in California was selected as it had a PV array nearly equal in size to that of Amherst and was located on a similar latitude. The remaining cities in California were selected based on the subsequently decreasing capacities of the PV arrays located at the given cities.

Although multiple years of data were obtained from each source and location, there were no overlap of years between the site in Massachusetts and those in California. The University of Amherst accrued historical data for the power output of their array with a 15-minute refresh rate from the years 2017 up until the current day when this was written. The output power from Amherst was compiled from the year 2018 to 2020. This contrasts with the years available from SIG Energy. Although the data also contained a 15-minute sampling frequency, it was only available from 2011 to 2016. For each year of data available, the dataset followed the calendar year by starting on January 1st and ending on December 31st. Fortunately, despite this discrepancy, the weather data maintained a consistent dimension of parameters for each location. When conducting cross-city analyses on three years of data, all five cities were used. Therefore, only the cities in California were used when conducting cross-city projections for six years. To reduce any potential differences that may arise when comparing the datasets from California and Massachusetts, only the last three years of the SIG Energy data were utilized for the cross-city analyses across the five cities.

The datasets underwent preliminary preprocessing, prior to any standard preprocessing required for the algorithms. This preliminary preprocessing predominantly consisted of filtering the dataset to match the daily window of 10am to 3pm, composed of 11 data points per day. However, in addition to that, the output power from SIG Energy was converted from kWh every 15 minutes to kW per 30-minute window. The datasets for each city are composed of 22 columns, wherein the first five are the date and time, the next 16 are the weather data, and the last column is the output power. For the datasets of three years in length, there are 12,056 rows while the datasets of six years in length are twice that at 24,112 rows. The columns and the respective units are described in more detail in the bulleted list.

2.2 Data Description

- **Year**: For the data at Amherst, the years range from 2018-2020, while the Californian cities range from 2011-2016. Wherein only the last three years from the Californian cities are compared to those from Amherst.
- **Month**: All twelve months of data are included for each year of data available.
- **Day**: Each day, including leap days, was included.
- **Hour**: Only the hours from 10am to 3pm and subsequent data points within were used.
- **Minute**: The data was sampled at a 30-minute sampling frequency.
- **Diffused Horizontal Irradiance (DHI)** $[W/m^2]$: The solar radiation that has indirectly arrived at a given location after having been scattered by the clouds and other particulate matter in the

atmosphere. The solar radiation arrives equally from all directions [21].

- **Diffused Normal Irradiance (DNI)** [W/m^2]: It is representative of the total radiation that arrives perpendicular to a given surface, and is measured by photoelectric detectors [16].
- **Global Horizontal Irradiance (GHI)** [W/m^2]: It is representative of the total amount of shortwave radiation that is received from the sun by a given horizontal surface. The GHI was measured by thermoelectric detectors and can also be calculated from the equation: $GHI = DNI * \cos(SolarZenithAngle) + DHI$
- **Clear sky (DHI)** [W/m^2]: Clear sky denotes the conditions where there is an absence of clouds across the entire visible sky. The clear sky DHI is therefore the upper threshold that the DHI could achieve.
- **Clear sky (DNI)** [W/m^2]: The clear sky conditions for DNI.
- **Clear sky (GHI)** [W/m^2]: The clear sky conditions for GHI.
- **Cloud Type [Unitless]**: NREL classifies clouds into eleven types. In increasing numbers, these types are probably clear, fog, water, super-cooled water, mixed, opaque ice, cirrus, overlapping, overshooting, unknown, and dust. Therefore, this specific column contains discretized data rather than continuous.
- **Dew Point [°C]**: The temperature at which water vapor can condense at. At the dew point, a saturation of water vapor is reached, therefore fog, clouds, and precipitation may develop [22].
- **Solar Zenith Angle [Degrees]**: The angle is the angle that the sun is relative to an axis that is normal to a surface [23]. This angle decreases as midday approaches, reaches a minimum value, then increases afterward. It equals the latitude minus the angle of solar declination.
- **Surface Albedo [Degrees]**: The fraction of solar radiation that is reflected by the surface of the Earth [24]. This value varies between zero and one, whereby a higher value indicates a larger amount of radiation that was reflected off the Earth.
- **Wind Speed** [m/s]: The speed of the wind within the atmosphere. This measurement was taken at the surface and scaled to within the atmosphere [16].
- **Precipitable Water [mm]**: The cumulative amount of water vapor contained within a vertical column of a given space of the atmosphere. The volume of water vapor is typically expressed as if it had condensed.
- **Wind Direction [Degrees]**: The direction is determined by the nearest hourly values of the second iteration of the Modern Era Retrospective analysis and Research Applications [16].
- **Relative Humidity** [%]: The total amount of water vapor in the air compared to the maximum vapor that the air can retain at a given temperature [22].
- **Temperature [°C]**: The temperature is measured at surface level but scaled to be the temperature

within the atmosphere by a factor of 6ºC/km of elevation increased.

- **Pressure [mbar]**: The measured pressure of the atmosphere.
- **Output Power [kW]**: The power generated by the PV array. The upper threshold of this value is dependent on the size of the array, while the instantaneous value is dependent on the weather factors above.

3 Machine Learning Methods

There were numerous models whose accuracies were compared in this study. Due to the similarities between select models, they have been grouped into families. These algorithms can be classified as distance-based, linear, kernel, ensemble methods, and neural networks. The numerous regression models were compared to determine which model, or set of models, performed best on the dataset accumulated.

3.1 Distance-Based Model

The KNN algorithm utilizes a type of instance-based supervised learning, based on the differences between features. The algorithm uses the distance function to determine a set of samples, whose length is dictated by the value of k, that are the closest to the target variable [25]. The algorithm stores the entire training dataset during the training phase. The model then creates a set of instances of length k that most closely maps to the target. The prediction of the model is created based on the similarity that the new observations have with the aforementioned set formed during training. These new instances are compared to each instance within the training set, the prediction is derived from the average of the response variable [25]. In regression based KNN, the algorithm computes the prediction Y for each instance of x by averaging the targets from the nearest k instances from the set, as described in Eq.(1).

$$Y = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{1}$$

where in this simplified example, x_i represents the training examples, and $N_k(x)$ is the set of nearest points [25]. It can be difficult to determine the optimal value of k as there is an inverse relationship between k and the error on the training set but a direct relationship with the error on the test set. The distance function, used to calculate the Euclidean distance d between the variables x and y is used in the KNN algorithm as described in Eq. (2).

$$d_{x,y} = \sqrt{\sum_{i=1}^{n} x_i - {y_i}^2} \tag{2}$$

This model differs from the others from its simplistic nature by using distance-based weighting for the samples closest to the target. Because of this functionality, it tends to be an ideal model to run on a dataset early into the researching phase.

3.2 Linear Models

There are numerous types of models that fall within the parent category of linear regression. In this study, a few of these algorithms that were compared include the baseline linear regression, linear regression with stochastic gradient descent as an optimizer, PLS, ridge regression, kernel ridge, and elastic net.

Linear regression is one of the simplest models that could be tried when conducting regression on a dataset, therefore acting as a baseline performance. As shown in Eq. (3) the correlation between the independent variable x and the dependent variable y is bridged with a weighed coefficient for each dependent variable and an intercept [26]. The objective of a linear regressor is to determine the values of w and $w[0]$ such that the loss function is minimized. Furthermore, the loss function must be defined within the predicted and actual values of the target variable [27].

$$y = w[0] + w[1] * x[1] + \ldots + w[n]x[n] \tag{3}$$

where y is the output, or dependent variable, $x[1]...x[n]$ are the independent features, $w[1]...w[n]$ are the coefficients of the linear model and $w[0]$ the intercept term. A variation of the baseline linear regressor is also compared in this study. Linear stochastic gradient descent regressor is a linear regressor that uses stochastic gradient descent as an optimizer. This model iteratively updates the model weights using a small, randomized subset of the training data instead of the entire dataset. Therefore, making it computationally efficient for larger datasets [27]. Although as the complexity of a dataset increases, the likelihood of a linear regressor producing accurate projections tends to decrease.

Another variation of a linear regressor is PLS. It differs from the previous regressors as it is based on covariance, and is often employed in circumstances where there are many independent variables where some are correlated. PLS reduces the number of variables to predict a smaller set of predictors [28]. This smaller set is then used to perform the regression analysis. Although there are two types of PLS for regression, PLS 1 and PLS 2, the applications for each differ if there are one or multiple dependent variables, respectively. Given there was only one dependent variable in this study, PLS 1 was used [28]. The formula for PLS regression is described in Eq. (4).

$$\begin{aligned} &Y = ThC'h + Eh = XWh * C'h + Eh = XWh(P'hWh) - 1C'h + Eh, \\ &B = Wh(P'hWh) - 1C'h \end{aligned} \tag{4}$$

where Y is the matrix of dependent variables, X is the matrix of independent variables, B is the matrix of regression coefficients generated by PLS of Y on X with h number of components. While Th, Ch, $W * h$, Wh, Ph are the matrices generated by the algorithm, and Eh is the residual.

Ridge regressors are algorithms that estimate the coefficients of multiple regression models where the independent variables are highly correlated. What separates ridge regression from PLS is how the model handles the multicollinearity. Multicollinearity occurs when any two independent variables are correlated [29]. Instead of utilizing covariance matrices, ridge uses L^2 regularization is to minimize the penalized sum of squares to yield the weighted ridge coefficient. Ridge is typically used when the independent and the dependent variables have been centered [30]. The process of determining the ridge coefficient is described in Eq. (5). Where y is the dependent variable in the regression model, λ is a a small coefficient, X is the design matrix, and I is an identity matrix [30].

$$\hat{\beta}_{ridge} = \left(XX^T + \lambda I\right)^{-1} X^T y \tag{5}$$

Elastic net is a sparse learning regressor that solves the limitations of the Lasso and ridge regressors yet maintains both as special cases. In Lasso regression, the independent variables are shrunk to a central value, eliminating irrelevant parameters and utilizes the L^1-norm [31]. This contrasts with ridge regression that utilizes the L^2-norm and minimizes the impact of the irrelevant parameters. Elastic net employs a weighted combination of the L^1 and L^2 regularization methods used by the component algorithms. This algorithm is able to generate reduced models by creating zero-valued coefficients [32]. Elastic net is often preferred, as it can apply the optimal regularization technique based on the nature of the data. As a result, it is considered to be a parent model to Lasso and ridge regression [31]. Elastic net is described further in Eq. (6)

$$min_{P_\alpha(\beta)} = \left(\frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \beta_0 - x_i^T\beta\right)^2 + \lambda P_\alpha(\beta)\right),\ P_\alpha(\beta) = \sum_{j=1}^{p}\left(\frac{1-\alpha}{2}\beta^2 + \alpha|\beta_j|\right) \tag{6}$$

where N is the number of observations, y_i is the response at observation i, x_i is the data as a vector of p values at observation i, λ is a positive regularization parameter, $P_\alpha(\beta)$ is the penalty term, α is a scalar that ranges between zero and one, and β_0 and β are scalars [32]. When α equals one, elastic net applies L^1-norm and functions like Lasso regression, alternatively, as α approaches zero, elastic net approaches the L^2-norm, therefore functioning comparable to ridge regression. If the elastic net is operating similarly to ridge regression, then the algorithm would use gradient descent to generate the projections. If elastic net is either completely or partially configured to operate as Lasso regression, then subgradient descent or coordinate descent would be used. In the case where α is between zero and one, then both L^1- and L^2-norm would be used by the algorithm. Due to the robustness of elastic net, it was one of the models whose results were displayed graphically.

3.3 Kernel Methods

A variation of the standard ridge regressor is kernel ridge, which combines the least squares and L^2-norm of ridge regression with the kernel trick. The kernel trick means that the computations for all pairs of data in the feature space are calculated from the inner products of vectors [33]. The usage of the kernel trick sets kernel ridge apart from the ridge regressor. Additionally, the usage of the squared error loss function differentiates it from SVR. In this study a polynomial kernel of degree ten is employed, thereby mapping the function to the original space [33].

Support vector regression is an abstracted version of support vector machines. SVRs are better suited for times-series predictions, which are the conditions that governs forecasts for PV power generation when using irradiance. In a more general sense, the SVR is derived from a function that maps the input patterns to those of the output. This is done based on a given set of training data that aims to minimize error by individualizing the hyperparameters. The input features are mapped using a non-linear mapping process to a high-dimensional space [11]. The nature of the SVR can is described in Eq. (7).

$$y(x, w) = \sum_{i=1}^{N} w_i k(x, x_i) \tag{7}$$

where $\{x_i, t_i\}_{i=1:N}$ is the training set and many of the w_i's are equal to zero. However, there are some limitations to the SVR algorithm: it lacks probabilistic interpretation, there is difficulty in selecting the optimal regularization parameter C, and the algorithm is restricted to using positive semi-definite kernels [34]. The projections of NuSVR were also compared in this study. *Nu* is a parameter used to control the number of support vectors, replacing the parameter epsilon in epsilon-SVR [35]. In this case, a *nu* value of 0.35 was used. For both SVR and NuSVR the radial basis function kernel was used as the activation function. The NuSVR algorithm was selected to represent this group of models given it had a more robust performance overall.

3.4 Ensemble Models

The basic principle behind ensemble methods is to create an integrated group of baseline models, typically considered weak learners, into a more robust model [10]. Where the more robust model can better adapt to changes in the dataset, thereby providing more accurate and reliable performances regarding the projections. Although varying degrees of ensemble models were used in this study, all of them were composed of regression trees.

In general, tree-based regression models benefit from a simpler structure and higher efficiency when applied to large domains of datasets. This is a result of the fast divide-and-conquer behavior of the models. Furthermore, regression trees are based on the greedy algorithm, wherein the larger dataset is split recursively into smaller partitions [36]. Although these tree-based algorithms are effective for large datasets, they prove to have shortcomings such as instability on smaller datasets [36]. This instability could arise from a small change during the training phase leading to different nodes being created and subsequently causing said instability and inconsistent results.

The weakest learner of these tree regressors are decision trees. Decision trees are composed of the potential decisions and corresponding repercussions, constructed in a flowchart-like tree structure [25]. The outcome of a node is represented by the branches or edges. Each node either has a decision node, chance node, or end node. A Boolean argument is representative of the branches or edges. The decision tree then weighs the three aforementioned conditions [25].

Random forest is a type of supervised learning algorithm that effectively uses ensemble bagging to tackle regression- or classification-based problems. During the training phase, the algorithm creates multiple decision trees and then outputs the mean prediction of the trees [37]. The benefit of having multiple trees, is that the collection of trees protects against the errors of the individual counterparts. The random forest model acts as an aggregator to the mean projections of the total decision trees constructed [37]. This architecture of aggregating multiple decision trees is displayed in Fig. 2. In this study, the tree algorithms use squared error as the loss function.

Ensemble models are characterized based on the method that was used to build the weak learners. There are three types of ensemble methods that are typically used: bagging, boosting, and stacking [10]. The term “bagging” was derived from bootstrap aggregation. It is where multiple baseline models are trained in parallel on portioned subsets of the training data. During the training phase, bootstrapping occurs, where the original dataset is randomly sampled with replacement. Sampling with replacement means that every time a sample is collected by a model, it is then replaced [38]. This ensures that each round of sampling is independent and does not interfere with the next round. Then, the final prediction of the algorithm is obtained from a voting aggregation of the final predictions

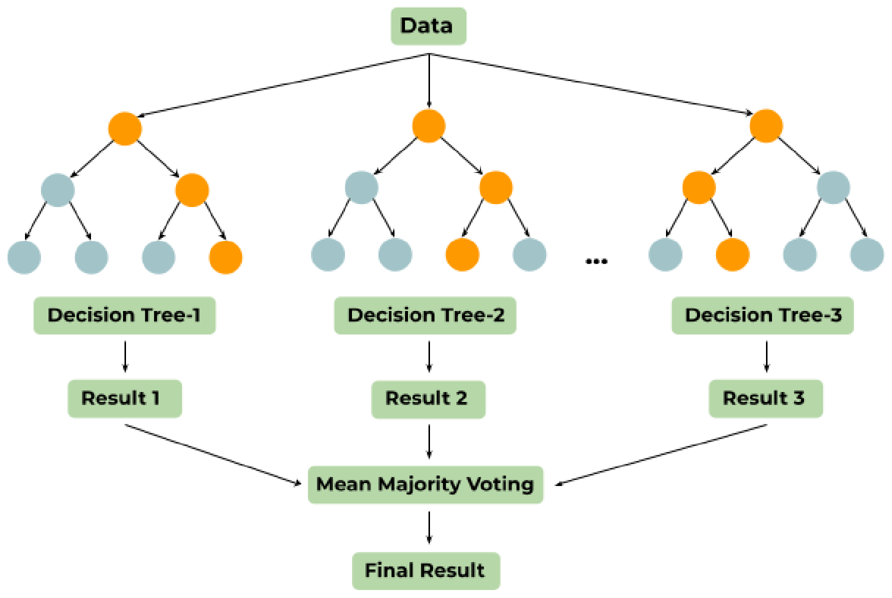


Figure 2: The architecture of a simple random forest.

of the baseline models [10]. Given that random sampling with replacement is used within ensemble methods, instead of altering the biases of the models, the variance of the projections is reduced. In this study, a version of ensemble bagging composed of random forest models was utilized. Therefore this was equivalent to the aggregation of final result from multiple random forests as displayed in Fig. 2. Due to the nature of the ensemble bagging model, it is a particularly strong algorithm and is the best of the machine learning models compared in this study.

3.5 Neural Networks

These neural networks are composed of the three typical layers such as the input, hidden, and output layers. Although the size of the output layer remains constant across each of the models, the size of the input and hidden layers vary. The implementation of the MLP, differs more significantly from that of the LSTM and GRU. Once such difference is that the architecture of the MLP more closely resembles that which is displayed in Fig. 3, with one hidden layer. This contrasts with the LSTM and GRU which have multiple inter-connected hidden layers. These multiple hidden layers can be conceptualized as another column of hidden neurons. In Fig. 3, x_m represents the number of features of the input layer, h_n the number of hidden neurons, and $\hat{y}$ is the target variable.

The LSTM and GRU algorithms are subsets of RNN and were developed years ago to combat the vanishing gradient problem, that is typical of RNNs [39]. The vanishing gradient becomes prevalent during in RNNs that utilize gradient descent learning. The vanishing gradient occurs when the partial derivative assigned to a weight is so small that the weight is effectively eliminated [40].

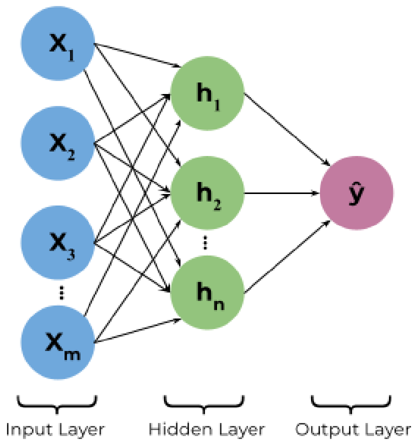
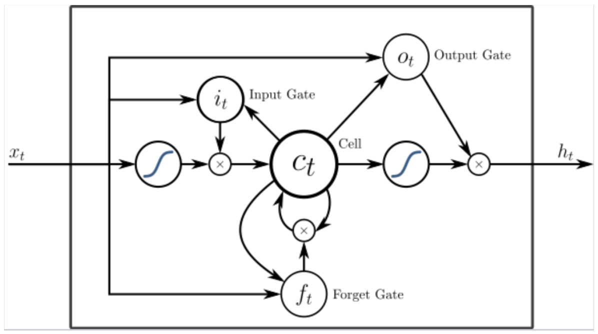


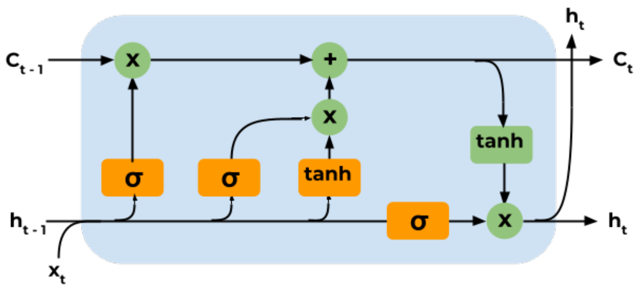
Figure 3: The architecture of a simple neural network.

MLP is simple neural network with a feed forward supervised learning algorithm. Within this model a non-linear activation function is used when each neuron in the hidden layer transforms the previous dimensions of the input layer using a weighted linear summation [41]. Additionally, back-propagation is used without the need of an activation function in the output layer, effectively using the identity function as an activation function. For the forward propagation in this study, the rectified linear unit activation function is used. Additionally, the Adam optimizer, an extended version of stochastic gradient descent optimizer, and the square error loss function were implemented into the MLP. Furthermore, the MLP was implemented such that during the training and testing phases, the algorithm took in the entire dataset from the respective phases. The MLP algorithm is beneficial as it has the capability to learn non-linear models as well as learning models in real time. Although, the hidden layers have a non-convex loss function, which leads to the potential for multiple minima to exist [41]. Therefore, any differences in the random weighting of the initialization can cause more significant variations in the accuracy of the validation. Additionally, the MLP is subject to sensitivity when feature scaling.

Unlike the MLP, the LSTM network consists of a series of gating mechanisms. This network is composed of several types of gates that contain information about the previous state. The information of the LSTM is either written, stored, read, or eliminated in the cells that serve as a memory stage for the model [42]. The four potential processes are accomplished through the selective opening or closing of the gates. The cells act on signals they receive and based on the strength of the signal they will either transmit or block information. The LSTM model is composed of three different states, the input, hidden, and output state. Within each unit of the LSTM there exists a cell state, C_t, an input gate, i_t, an output gate, o_t, and a forget gate, f_t, displayed in Fig. 4. The forget gate is tasked with determining which information is kept or eliminated from the cell state [42]. This decision is



(a) The LSTM Unit. Source [42].



(b) The LSTM Cell.

Figure 4: The LSTM unit with the forget, input, and output gates, while the LSTM cell is able to process data sequentially and maintain hidden states through time steps.

determined by the logistic function f_t, as described in Eq. (8). Where this function will either output a value of zero, to keep the information, or a value of one to forget it.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{8}$$

where, in equations (8) and (9), σ is the activation function, W_f is the weight of the forget gate, b_f is the bias of the forget gate, x_f is the input at time t, h_{t-1} is the hidden layer at time $t-1$, W_c is the weight of the cell, and b_c is the bias of the cell. The input gate, forget gate, cell state, and output gate are shown in Fig. 4, the LSTM cell. The input gate, i_t, and the cell state, C_t, are described in Eq. (9). The input gate determines which input values are updated by the blocks of the LSTM.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i),\ C_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{9}$$

The output state determines which segment of the cell state is permitted to output. The formula for the output state, as described in Eq. (10) includes *tanh* and is multiplied by another logistic function whose output is scaled similar to the forget state.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o),\ h_t = o_t + \tanh(C_t) \tag{10}$$

where σ is the activation function, W_o is the weight of the output gate, and b_o is the bias of the output gate. The input data to the LSTM is composed of a three-dimensional array. The first dimension is represented by the number of samples in the network, the second dimension is the time-steps, and the third is the number of features in one input sequence [42]. In order for the LSTM to properly handle the dataset, a sliding window was created. The implementation of the LSTM and GRU by utilizing the sliding window separate these models from the other algorithms evaluated in this study. The sliding window is discussed in greater detail in the preprocessing section. That said, the resulting size of the three-dimensional array inputted into the LSTM was 11 by 3 by 20. The version of the LSTM in this study contains a batch size of 64, a hidden size of 64, three dropout layers, and a MSE loss function used given the regressive nature of the dataset. Additionally, the Adadelta optimizer was determined to yield the best performance by trial-and-error. The Adadelta optimizer is a more robust version of the Adagrad optimizer. Adadelta adapts the learning rates based on a moving window of gradient updates, therefore, it is not necessary to set an initial learning rate [43].

Although the GRU shares similarities to the LSTM in terms of the functionality, there are still differences in the architecture nature of the models. Similar to the LSTM, the GRU is able to handle sequential data such as time series, speech, and text [39]. The GRU uses gating mechanisms to selectively update the hidden state, subsequently updating the output layer. One way that the GRU differs is that the model has two gates instead of three and lacks a cell state. The gating mechanisms of the GRU are composed of the update gate and the reset gate [39]. In the GRU model, the reset gate determines how much of the previous information of the hidden state should be forgotten. The reset gate of the GRU is analogous to the input and forget gate of the LSTM [40]. The update gate determines how much of the previous information should update the hidden state, and then subsequently be passed into future units of the algorithm. The update gate is comparable to the output gate of the LSTM. Within the reset gate exists another gate that is a subset of it. This is the current memory gate, and introduces non-linearity into the input data. The current memory gate is able to reduce the impact that the previous information has on the current information, which would be transmitted to any future units [40]. The GRU cell is displayed in Fig. 5. Where h_{t-1} is the previous state, x_t is the input, and h_t is the output. The final output of the GRU model is calculated based on the hidden state and is described in Eq. (11)

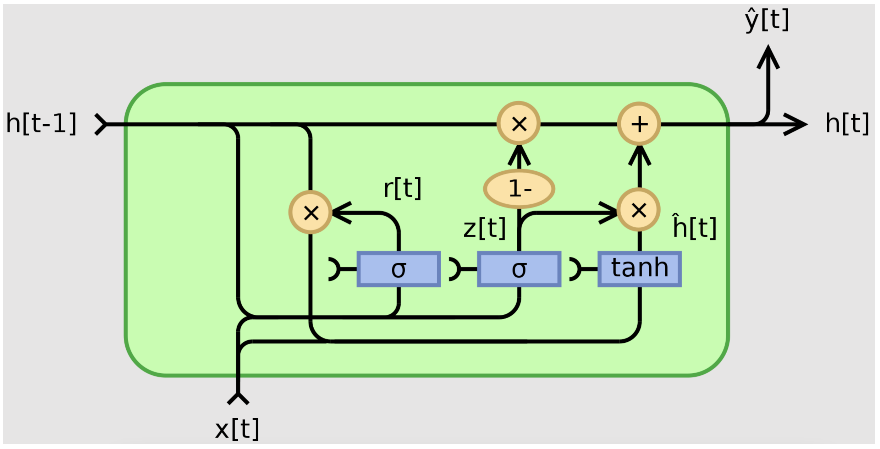


Figure 5: The structure of the GRU cell.

$$r_t = sigmoid(W_r[h_{t-1}, x_t]),\ z_t = sigmoid(W_z[h_{t-1}, x_t]),$$
$$h'_t = tanh(W_h[r_t h_{t-1}, x_t]),\ h_t = (1 - z_t)h_{t-1} + z_t h'_t \tag{11}$$

where r_t is the reset gate, z_t is the update gate, h'_t is the candidate hidden state, h_t is the hidden state, h_{t-1} is the prior hidden state W_r and W_z are the learnable weight matrices, and x_t is the input at time step t. The sigmoid function is applied to scale the result between zero and one. The GRU model is able to solve the vanishing gradient by storing the relevant information from one time step to the next of the network [39]. The GRU used in this research shared the input dimensions of the LSTM because the same sliding window was employed. Additionally, the model utilized an averaged stochastic gradient descent optimizer with a MSE loss function.

4 Experiments

4.1 Preprocessing

The research that was conducted included machine learning and deep learning models. To improve the accuracy of the projections of these models feature selection was conducted on the dataset. The parameters of interest were determined from the generation of Kendall correlation heatmaps. Additional preliminary testing was conducted to evaluate the optimal normalization technique to be used on the compiled dataset. The normalization methods that were compared were min-max, z-score, and decimal scaling. These methods were evaluated based on the accuracy of the models given same-city and cross-city projections when trained at Huron. Ultimately, min-max scaling was used for the evaluations within the study. The data was then either split sequentially or randomly depending on the type of model used, although in both cases there was a 70/30 split of training data to testing data. Sequential data was used for the deep learning models of LSTM and GRU, while all the other models underwent random sampling. The machine learning models were inherited from the scikit-learn library. Based on the input requirements for the machine learning models, the entire training dataset was inputted in one instance prior to testing. This is in contrast to LSTM and GRU algorithms. The two deep-learning models were trained and tested using a sliding window. The sliding window dropped the year but retained the other independent parameters of data and was composed of 11 rows such that the equivalent of one day of data was used in each instance. The output power data from the 11th row was the dependent variable tied to the entire sliding window. The sliding window and the respective output power value were saved as a pickle file. The hyperparameters of each model were tuned through trial and error.

The experimentation remained the same for each model. The models were trained on the dataset from one city, then tested three times on the dataset of a given city. The location of the testing city was cycled after recording the results of the evaluation metrics from the mean and standard deviation of each iteration. The process was then repeated by cycling through which city was used as the training dataset. The same-city projections and cross-city projections results, from datasets of a given length of years, were extracted from the large matrix of results generated.

The above process was conducted for the five cities with datasets of three years in length and then repeated for the four cities with six years of data. Varying the length of years of data inserted into the models allowed for deeper evaluations of the performances. Namely, to observe if there was any fall-off of accuracy for certain models.

4.2 Evaluation Metrics

Although there is no standardized evaluation metric applied to determine the accuracy of forecasts for the output power of PVs, there are a few that are more commonly used. The acceptance of using MSE or RMSE varies from one report to another, as there are numerous metrics used within each study. Occasionally R^2 is used as well. R^2 is a statistical measurement of how well the regression coefficient matches the ground truth. For R^2 scores, the closer a score is to one, the more accurate that projection, while smaller decimals and negatives are indicative of increasing inaccuracies. This is in contrast to the ideal MSE and RMSE values, where the more accurate the projection of a model is, the closer the value would be to zero. The results from MSE and RMSE are strictly positive.

At least in regard to electricity forecasting, RMSE appears to be a popular evaluation metric [44]. The three accuracy measurements of R^2, MSE, RMSE, were used to better enable comparisons to prior contributions. The evaluation metrics used are described in Eq. (12).

$$R^2 = 1 - \frac{\sum y_i {-_i}^2}{\sum y_i - \tilde{y}^2},\ MSE = \frac{1}{n}\sum_{i=1}^{n} y_i - \tilde{y_i}^2,\ RMSE(y_{T+1}, \tilde{y}_{T+1}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i - \tilde{y_i}^2}, \tag{12}$$

where n is the number of iterations, y_i is the observed value, and $\tilde{y_i}$ is the predicted value.

4.3 Experimental Results and Analysis

In any dataset, particularly in datasets with many dimensions, it is beneficial to conduct feature selection. Feature selection is a technique employed to improve the accuracy of projections by removing the multicollinearity in independent variables. Multicollinearity between two variables can be easily identified when the magnitude of the correlation coefficient is equal to or exceeds 0.7 [29]. This is shown in Fig. 6, where a cross-correlation heatmap was generated based on the Kendall coefficient. The Kendall correlation was selected as it is both more robust and ideal for time series data because it can handle normally and non-normally distributed data. When interpreting heatmaps, the learned

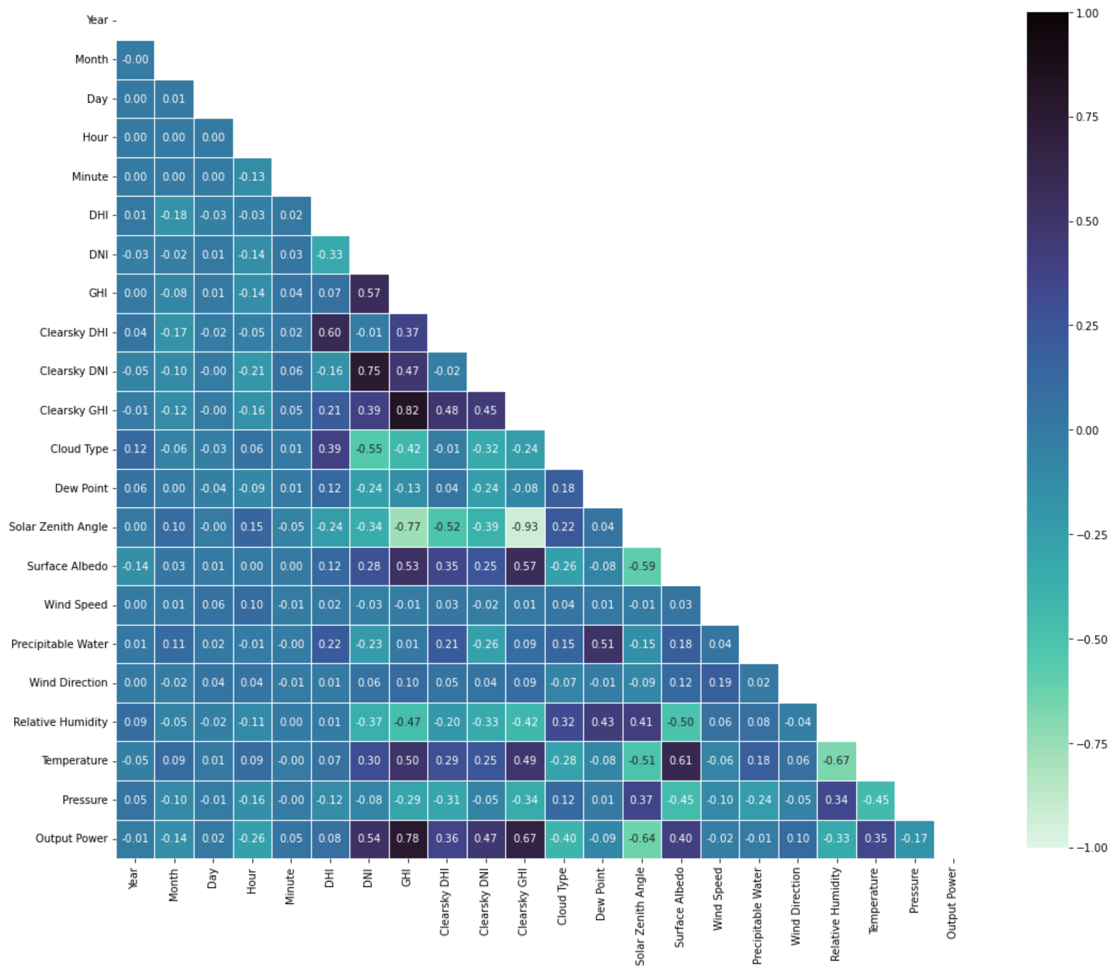


Figure 6: The Kendall cross-correlation of the parameters of the dataset.

weight that a given parameter on the horizontal axis has on a parameter on the vertical axis is given by the value at the point of intersection of the two parameters. If the value of the coefficient is greater than zero, there is a positive relationship between the two parameters, while if the coefficient is less than zero, then there is an inverse relationship between the parameters. And if the coefficient is equal to zero, then there is no correlation.

Although there is a slight variation in the value of the weight coefficients, based on the city used in the cross-correlation, the relative ratios between the weights remain similar. There are a few cases of multicollinearity within Fig. 6. This occurs between GHI and clear sky GHI, DNI and clear sky DNI, GHI and solar zenith angle, and clear sky GHI and solar zenith angle, with coefficient weights of 0.82, 0.75, -0.77, -0.93, respectively. According to the heatmap, the parameters with the greatest correlation with the output power of a PV array are GHI, followed by clear sky GHI, solar zenith angle, and DNI, with weights of 0.78, 0.67, -0.64, and 0.54, respectively. Unfortunately, the select variables that are highly correlated with the output power also exhibit multicollinearity with the other correlated

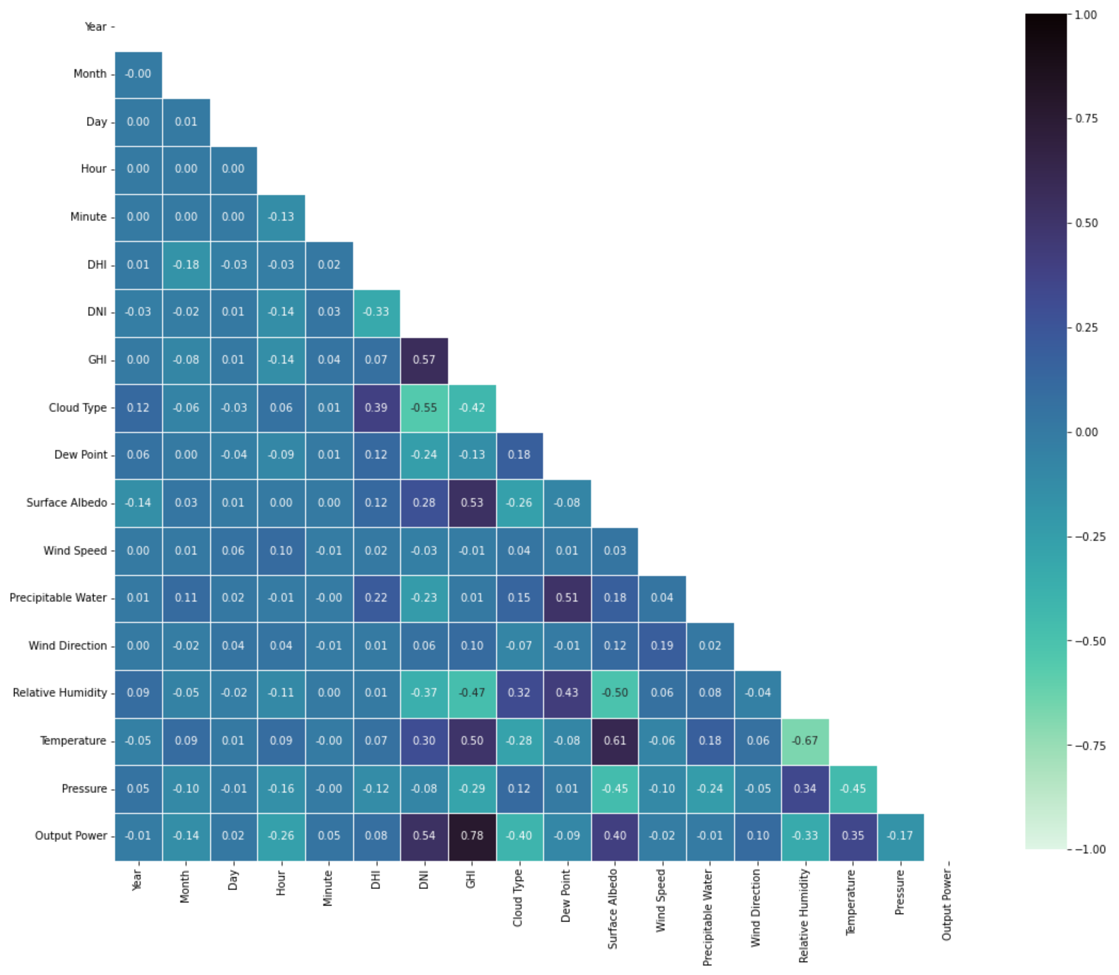


Figure 7: The Kendall cross-correlation after the dimensionality of the parameters were reduced.

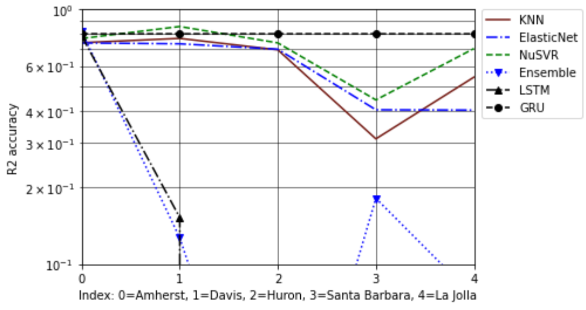
predictor variables. To remove the multicollinearity within the dataset, the dimensions of solar zenith angle, clearsky DHI, clearsky DNI, clearsky GHI were removed. A secondary Kendall correlation was created as displayed in Fig. 7. However, the accuracy of the model projections decreased relative to before the dimensionality of the dataset was reduced. Therefore, the complete dataset was used for the remainder of the comparative analyses. Prior to the reduced dimensionality, the parameters that were highly correlated with the output power of PVs aligns with that of prior art. Logically, the more solar radiation that a given area receives as measured by GHI, the greater the power output of the array. The reason the weight coefficient for the solar zenith angle is negative is that as the sun rises in the sky, its angle from an axis normal to the ground decreases, indicative of an inverse relationship between solar zenith angle and output power. The high correlation between these parameters, with the output power aligns with the existing literature within the field.

A few normalization techniques were compared to determine the optimal method for the specific dataset utilized in this study. The RMSE accuracy generated by these methods were compared through same-city and cross-city projections as displayed in Table 1. The normalization techniques were min-

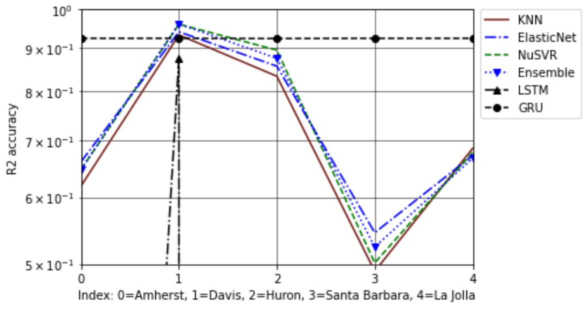
Table 1: The RMSE results from different normalization methods on three years of data.

	Same-City Projections			Cross-City Projections		
Model	Min-Max	Z-Score	Decimal	Min-Max	Z-Score	Decimal
KNN	0.082184	0.306474	0.000449	0.515207	0.513813	0.003769
Elastic net	0.061615	0.259507	0.001332	0.510282	0.484004	0.004285
NuSVR	0.046171	0.232047	0.000399	0.519203	0.488566	0.003741
Ensemble bagging	0.046672	0.221522	0.000357	0.499279	0.470452	0.003747
LSTM	0.077469	1.235138	1.215788	0.375896	1.261427	1.215734
GRU	0.074616	1.195654	1.157607	0.375316	1.225943	1.157613

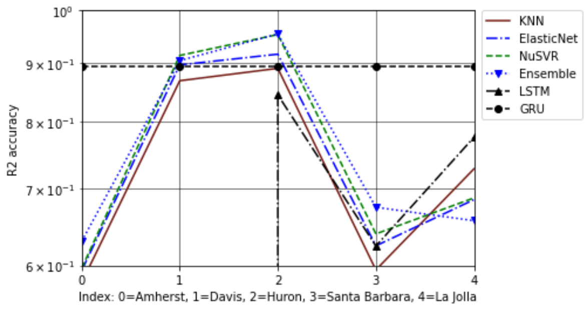
max, z-score, and decimal scaling. Min-max normalization maps the data to be within the range of zero to one by scaling each instance by the maximum and minimum of the dataset [45]. Z-score normalization, also named Zero-Mean normalization, is derived from dividing the difference of the instance and mean of the data by standard deviation of the data. While decimal scaling was performed



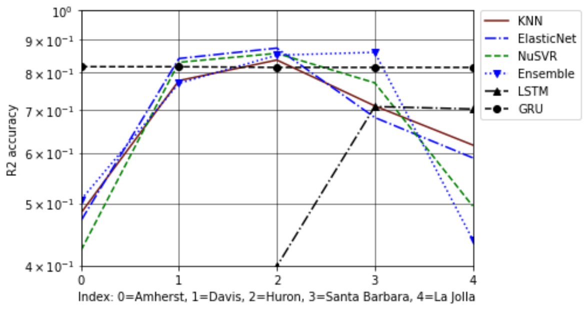
(a) Amherst Cross-City Projections.



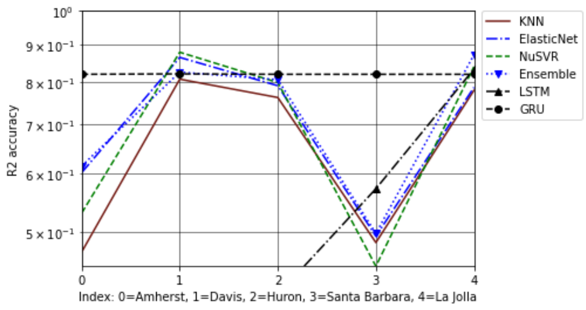
(b) Davis Cross-City Projections.



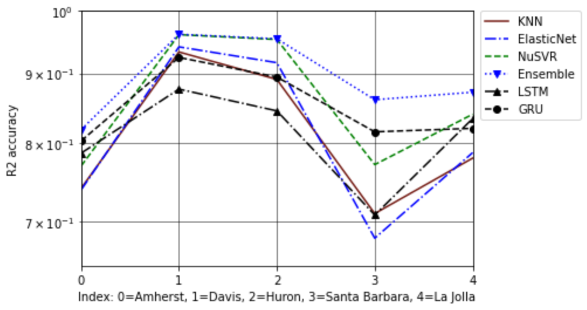
(c) Huron Cross-City Projection.



(d) Santa Barbara Cross-City Projections.



(e) La Jolla Cross-City Projections.



(f) Same-City Projections.

Figure 8: The cross-city and same-city R^2 scores from three years of testing, on a semilog scale.

by dividing each instance by ten to the power of four [45]. It was to the exponent of four because that was the maximum length of the instances in the dataset. Although decimal scaling yielded smaller RMSE results for the machine learning models, the accuracy decreased when applied to the neural networks. Z-score normalization yielded competitive results for cross-city projections for the machine learning models; however, lacked accuracy for the other testing conditions and models. This is in contrast to min-max normalization which yielded the best accuracies across all conditions. For these reasons, min-max normalization was considered the optimal normalization method.

Although numerous models were compared, only the more robust models from each of the groups were displayed graphically. That said, tables were generated based off the compiled results from each model. These tables are inclusive or more models than those in graphs.

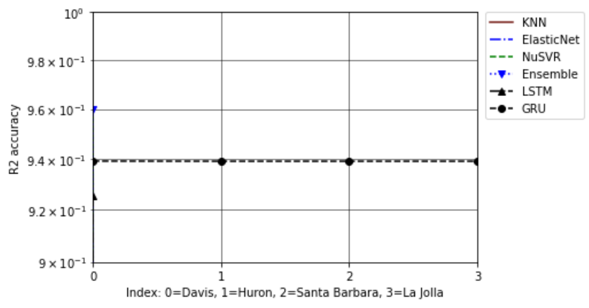
In general, the cross-city projections of models were less accurate relative to those of same-city projections. This was particularly evident for the R^2 evaluations, where numerous models had negative scores, in some cases this caused the results from select cities to not be graphed on a semi-log scale. These negative R^2 results indicate that the model's performance is less accurate than that of a constant function used to predict the mean [46]. The graphed R^2 scores from the different testing conditions across three and six years are displayed in Fig. 8 and Fig. 9, respectively. The discrepancies in the graphed results of cross-city projections for R^2 scores across six years of data are most likely caused by significant negative scores under certain testing conditions. Notably, GRU was the only model to have non-negative R^2 scores on six years of data, as seen in Fig. 9a the value averaged to be 0.94.

Table 2: The overall top performing models, given R^2 scores on three years of data.

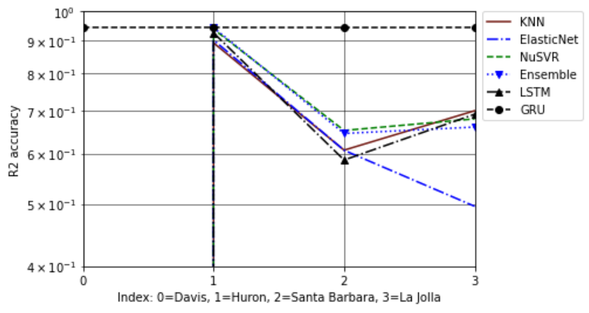
Training City	Same-City Projections	R^2 Score	Cross-City Projections	R^2 Score
Amherst, MA	Ensemble bagging	0.818208	GRU	0.802737
Amherst, MA	Kernel ridge	0.808784	NuSVR	0.701983
Amherst, MA	GRU	0.802711	MLP	0.630215
Amherst, MA	SVR	0.780876	KNN	0.611633
Amherst, MA	LSTM	0.786242	Elastic net	0.595406
Davis, CA	Kernel ridge	0.968095	GRU	0.924377
Davis, CA	Ensemble bagging	0.961323	Kernel ridge	0.748125
Davis, CA	NuSVR	0.960771	NuSVR	0.736539
Davis, CA	Random forest	0.958764	Elastic net	0.735150
Davis, CA	MLP	0.954446	Ensemble bagging	0.735079
Huron, CA	Kernel ridge	0.961908	GRU	0.894339
Huron, CA	Ensemble bagging	0.954075	Kernel ridge	0.765117
Huron, CA	NuSVR	0.952966	Ensemble bagging	0.763900
Huron, CA	Random forest	0.950676	NuSVR	0.758083
Huron, CA	MLP	0.946865	MLP	0.745354
Santa Barbara, CA	Ensemble bagging	0.860608	GRU	0.815995
Santa Barbara, CA	Random forest	0.842497	Elastic net	0.691191
Santa Barbara, CA	Kernel ridge	0.826094	Ensemble bagging	0.685700
Santa Barbara, CA	GRU	0.815155	KNN	0.684999
Santa Barbara, CA	MLP	0.796264	NuSVR	0.675440
La Jolla, CA	Ensemble bagging	0.871669	GRU	0.820216
La Jolla, CA	Kernel ridge	0.870437	Ensemble bagging	0.723362
La Jolla, CA	Random forest	0.863120	Elastic net	0.707779
La Jolla, CA	SVR	0.841832	Kernel ridge	0.707288
La Jolla, CA	NuSVR	0.839770	Linear SGD regressor	0.706491

Table 3: The overall ranking of the top performing models, given R^2 scores on three years of data.

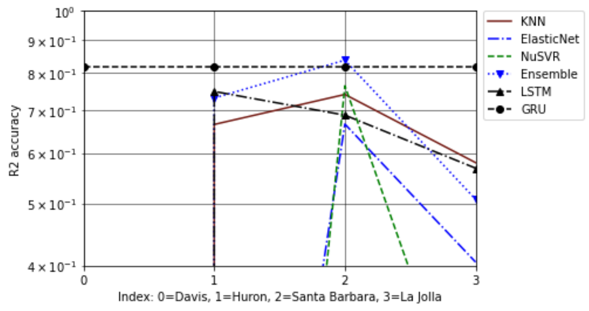
Model	Same-City Projections	Cross-City Projections
Ensemble bagging	23	11
Kernel ridge	21	10
GRU	5	25
NuSVR	7	10
Random forest	11	—
Elastic net	—	10
MLP	3	4
KNN	—	4
SVR	4	—
LSTM	1	—
Linear SGD regressor	—	1



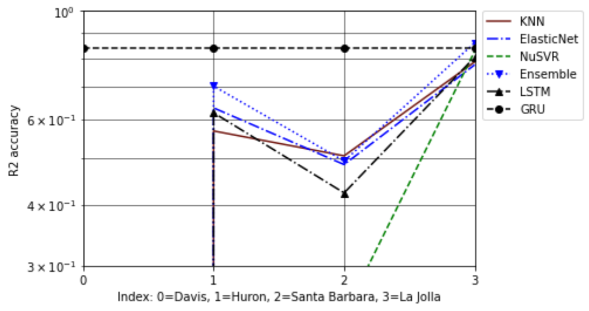
(a) Davis Cross-City Projections.



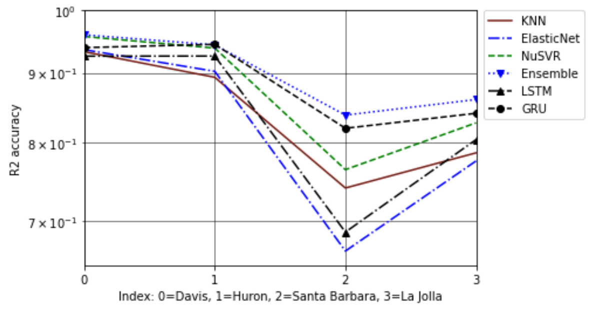
(b) Huron Cross-City Projections.



(c) Santa Barbara Cross-City Projections.



(d) La Jolla Cross-City Projections.



(e) Same-City Projections.

Figure 9: The cross-city and same-city R^2 scores from six years of testing, on a semilog scale.

Table 4: The overall top performing models given, R^2 scores on six years of data.

Training City	Same-City Projections	R^2 Score	Cross-City Projections	R^2 Score
Davis, CA	Kernel ridge	0.962847	GRU	0.939112
Davis, CA	Ensemble bagging	0.959678	SVR	-29.107155
Davis, CA	Random forest	0.957719	NuSVR	-35.422555
Davis, CA	NuSVR	0.956933	Linear SGD regressor	-37.781532
Davis, CA	MLP	0.954850	KNN	-37.950231
Huron, CA	Kernel ridge	0.948526	GRU	0.944224
Huron, CA	GRU	0.944319	NuSVR	-0.082058
Huron, CA	Ensemble bagging	0.943442	Ensemble bagging	-0.087932
Huron, CA	Random forest	0.938625	fandom Forest	-0.089763
Huron, CA	NuSVR	0.938625	Kernel ridge	-0.095945
Santa Barbara, CA	Random forest	0.848578	GRU	0.819383
Santa Barbara, CA	Ensemble bagging	0.848578	Random forest	-0.121863
Santa Barbara, CA	GRU	0.943939	Decision tree	-0.147946
Santa Barbara, CA	Kernel ridge	0.628883	KNN	-0.166443
Santa Barbara, CA	Decision tree	0.628905	Ensemble bagging	-0.190341
La Jolla, CA	Ensemble bagging	0.860190	GRU	0.840431
La Jolla, CA	Random forest	0.853891	Random forest	-0.144870
La Jolla, CA	Kernel ridge	0.849071	Decision tree	-0.160601
La Jolla, CA	GRU	0.840431	Ensemble bagging	-0.203896
La Jolla, CA	Decision tree	0.828522	KNN	-0.211623

Table 5: The overall ranking of the top performing models, given R^2 scores on three years of data.

Model	Same-City Projections	Cross-City Projections
GRU	9	20
Ensemble bagging	16	—
Kernel ridge	15	—
Random forest	14	—
NuSVR	3	—
Decision tree	2	—
MLP	1	—

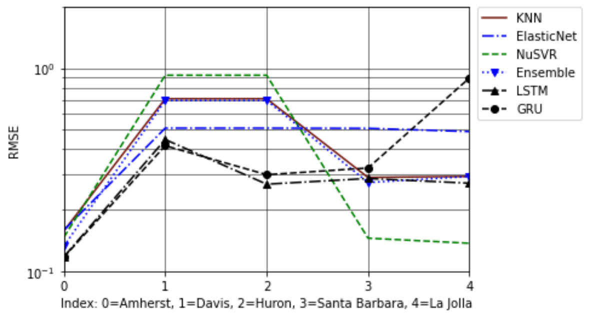
To assist in the comparison between the results from each model, the generated evaluation results were tabulated. Only the top five performing models of each testing condition were included in Table 2. In this study, the final value used in determining a model's cross-city performance was done by taking the average of all the results of projections including the same-city. While the same-city performance of a model was determined by obtaining the mean of the results from the training and testing phase occurring at the same location. For further readability, Table 3 was constructed based off Table 2. If a model was within the top five performances of any given testing condition for same-city or cross-city projections, then a score between one and five, inversely proportional to the ranking of the model, was assigned. If a model had the best performance, it received a score of five, while the 5th best model received a score of one. Subsequent models that did not perform within the top five models were not assigned any score. The models were then reordered based on the decreasing overall performances. When a model did not receive a ranking under any of the test conditions, it is reflected by a dashed line. The process of tabulating the results was repeated for R^2 scores across six years, and RMSE for three and six years, as shown in Tables 4 and 5, 6 and 7, and 8 and 9 respectively.

Table 6: The overall top performing models given RMSE results on three years of data.

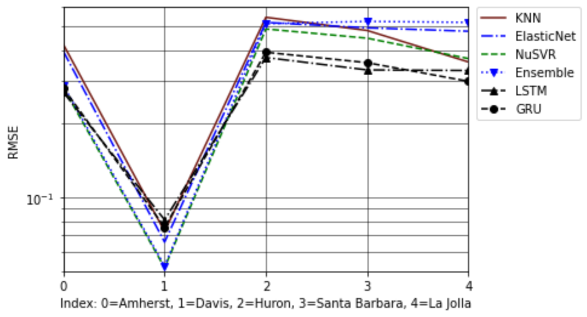
Training City	Same-City Projections	RMSE	Cross-City Projections	RMSE
Amherst, MA	GRU	0.117168	LSTM	0.277781
Amherst, MA	LSTM	0.117663	SVR	0.299399
Amherst, MA	Ensemble bagging	0.132658	Elastic net	0.433328
Amherst, MA	Kernel ridge	0.134039	Linear SGD regressor	0.434012
Amherst, MA	Random forest	0.140668	Random forest	0.421120
Davis, CA	Kernel ridge	0.045814	LSTM	0.277502
Davis, CA	NuSVR	0.046171	GRU	0.280577
Davis, CA	Ensemble bagging	0.046672	SVR	0.299829
Davis, CA	MLP	0.049761	NuSVR	0.326975
Davis, CA	Random forest	0.054414	KNN	0.376338
Huron, CA	Kernel ridge	0.041984	GRU	0.375316
Huron, CA	NuSVR	0.046171	LSTM	0.375896
Huron, CA	Ensemble bagging	0.046672	Decision tree	0.500686
Huron, CA	Random forest	0.048906	Elastic net	0.510282
Huron, CA	MLP	0.049761	Linear SGD regressor	0.510518
Santa Barbara, CA	Ensemble bagging	0.085397	LSTM	0.425840
Santa Barbara, CA	Random forest	0.093342	GRU	0.459018
Santa Barbara, CA	Kernel ridge	0.095554	Random forest	0.504005
Santa Barbara, CA	MLP	0.101864	KNN	0.505611
Santa Barbara, CA	NuSVR	0.107973	Ensemble bagging	0.509712
La Jolla, CA	Kernel ridge	0.074351	GRU	0.409212
La Jolla, CA	Ensemble bagging	0.075268	LSTM	0.438382
La Jolla, CA	GRU	0.076123	Ensemble bagging	0.551663
La Jolla, CA	Random forest	0.078357	Random forest	0.544993
La Jolla, CA	NuSVR	0.082536	Decision tree	0.545914

Table 7: The overall ranking of the top performing models, given RMSE results on three years of data.

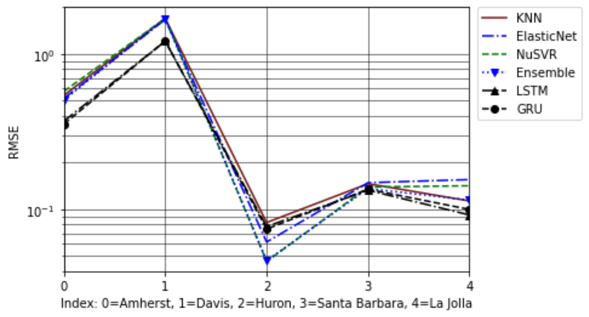
Model	Same-City Projections	Cross-City Projections
LSTM	4	23
GRU	8	18
Ensemble bagging	18	4
Kernel ridge	20	—
Random forest	10	6
NuSVR	10	2
SVR	—	7
MLP	5	—
Elastic net	—	5
Decision tree	—	4
KNN	—	3
LSGDR	—	3



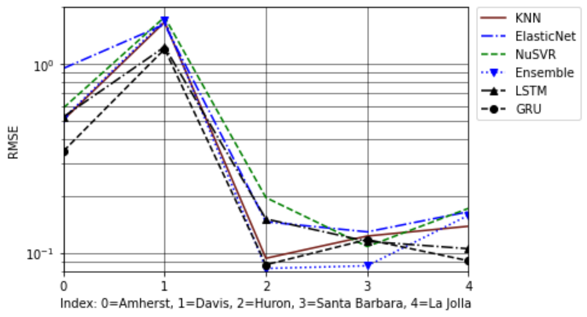
(a) Amherst Cross-City Projections.



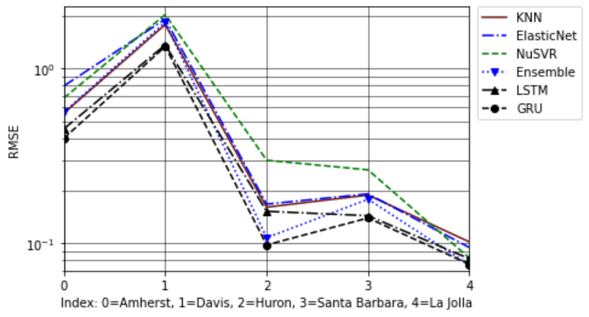
(b) Davis Cross-City Projections.



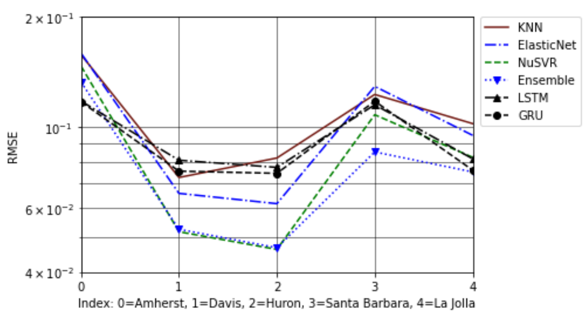
(c) Huron Cross-City Projections.



(d) Santa Barbara Cross-City Projections.

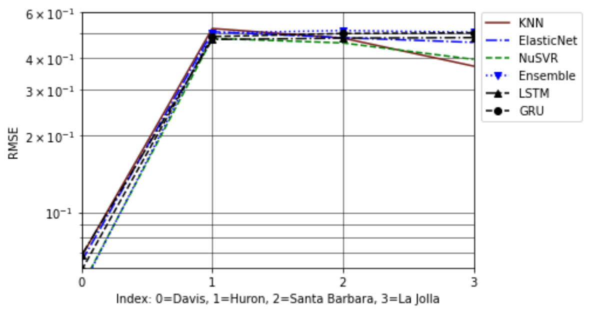


(e) La Jolla Cross-City Projections.

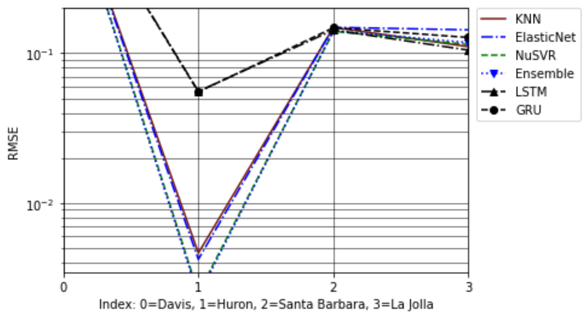


(f) Same-City Projections

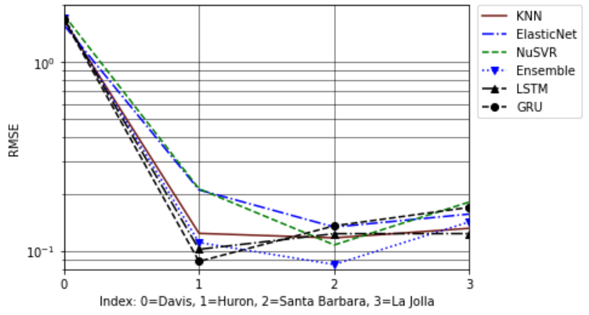
Figure 10: The cross-city and same-city RMSE results from three years of testing, on a semilog scale.



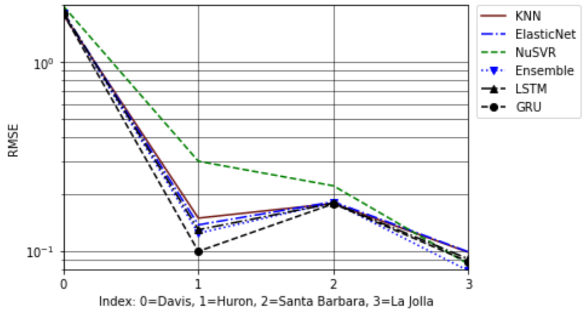
(a) Davis Cross-City Projections.



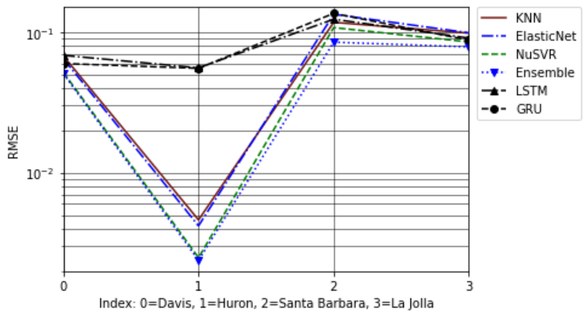
(b) Huron Cross-City Projections.



(c) Santa Barbara Cross-City Projections.



(d) La Jolla Cross-City Projections.



(e) Same-City Projections.

Figure 11: The cross-city and same-city RMSE results from six years of testing, on a semilog scale.

For the evaluation metric of R^2 scores, the models with the best performances remained relatively similar across datasets spanning three and six years. Given R^2 scores in Table 3 and 5, ensemble bagging generalized the best for same-city projections yet also had respectable cross-city forecasts. While GRU had the best projections for cross-city for both lengths of datasets. Based on the method for determining the cross-city R^2 performance of a model, GRU would be the only model that maintained accuracy across six years.

Although the accuracy metrics of MSE and RMSE were obtained, only RMSE is discussed in the report. This was because RMSE is more commonly compared between studies and was derived from MSE. LSTM was the most accurate model for cross-city projections for RMSE on three years of data, as shown in Table 7. Ensemble bagging performed the best given same-city projections, and GRU generalized the best for both conditions between the two models. The better performances of the

Table 8: The overall top performing models, given RMSE results on six years of data.

Training City	Same-City Projections	RMSE	Cross-City Projections	RMSE
Davis, CA	Kernel ridge	0.048115	SVR	0.317826
Davis, CA	Ensemble bagging	0.050737	NuSVR	0.344721
Davis, CA	NuSVR	0.051682	KNN	0.358947
Davis, CA	Random forest	0.052197	Linear SGD regressor	0.359327
Amherst, MA	MLP	0.053125	LSTM	0.373920
Huron, CA	Kernel ridge	0.002136	NuSVR	0.476343
Huron, CA	Ensemble bagging	0.002386	Ensemble bagging	0.477623
Huron, CA	NuSVR	0.002492	Random forest	0.477702
Huron, CA	Random forest	0.002576	KNN	0.478098
Huron, CA	MLP	0.002732	Kernel ridge	0.478177
Santa Barbara, CA	Ensemble bagging	0.084657	Random forest	0.498814
Santa Barbara, CA	Random forest	0.090800	Decision tree	0.505008
Santa Barbara, CA	Kernel ridge	0.099094	KNN	0.510331
Santa Barbara, CA	MLP	0.101365	Ensemble bagging	0.512817
Santa Barbara, CA	Decision tree	0.103732	LSTM	0.512968
La Jolla, CA	Ensemble bagging	0.078349	Random forest	0.537771
La Jolla, CA	Kernel ridge	0.080821	Decision tree	0.541429
La Jolla, CA	Random forest	0.080978	GRU	0.542811
La Jolla, CA	MLP	0.084491	Ensemble bagging	0.550890
La Jolla, CA	NuSVR	0.084968	KNN	0.551782

Table 9: The overall ranking of the top performing models, given RMSE results on six years of data.

Model	Same-City Projections	Cross-City Projections
Ensemble bagging	18	8
Random forest	11	13
Kernel ridge	17	1
NuSVR	7	9
KNN	—	9
Decision tree	1	8
MLP	6	—
SVR	—	5
GRU	—	3
LSTM	—	2
Linear SGD regressor	—	2

neural networks could be attributed to the memory storage of previous states inherent to these models. While the performance of the ensemble method could be attributed to the model's nature averaging the results across the baseline models relying on bootstrapping. Additionally, for RMSE results across six years, ensemble bagging had the best generalizations for both same-city and cross-city projections.

In this study, the kernel ridge model should be disregarded because the algorithm was not stable. It was common for the model to have cross-city MSE and RMSE results that were magnitudes worse when it was trained on any city and tested on Amherst or Davis. This is most likely caused by over tuning the model, as a 10th-degree polynomial kernel had been used.

From the extensive testing that was conducted, it became clear that select cities proved to be best for the training or testing phases. When conducting cross-city projections, the optimal city to use in the training phase was Davis. This is because when a model was trained on a dataset from a city that was not Davis, and tested on Davis, there were disproportionally large inaccuracies. Therefore, by being trained on Davis the cumulative inaccuracies were reduced. However, if the dataset from the city of Davis were to be omitted then the best dataset for the training phase for cross-city and same-city projections would be from Huron. Should this research be extended and applied to any given city, such that a model is trained on one city and tested on another, the dataset from the city of Huron should be used during the training phase.

5 Conclusion

Over the past few decades, the installation of PV arrays have increased exponentially. Although benefiting the environment by reducing the consumption of fossil fuels, this surge has primarily been driven by the reduction in costs of these RES. Given the ever-increasing implementation of PVs into electrical systems, and the intricate dependencies these sources have with shading and correlated weather conditions, it is critical that PVs are accurately modeled. Typically, select machine learning and deep learning algorithms have been adapted to model these RES with varied performances. The objective of this study was to compare the accuracy of a couple of neural networks that were implemented against machine learning algorithms to accurately model the output power of PV arrays. This testing was conducted on a dataset that spanned three and six years in length and was inclusive of five locations. It was determined that LSTM and GRU had the best overall performances, and of the two, GRU outplaced LSTM when considering R^2 scores as well. Although, if seeking an algorithm for a single location, ensemble bagging should be selected. That said, GRU was the most robust of the models when determining the output power of PV arrays in very short-term deterministic forecasts.

This application of cross-city projections could be expanded upon and used to model the PV arrays from any given city. This would be particularly beneficial to locations where limited data is available, yet a more extensive dataset available at a different location could be utilized to train the model. Should this application be further developed, the dataset from the city of Huron yielded the most accurate projections for modeling alternative cities and therefore could be used as a baseline. Additionally, these models and the cross-location projections could be applied to the modeling of other RES such as onshore or offshore wind farms.

References

[1] Maqsood A. Mughal. Machine learning. https://lucid.app/lucidchart/81100e02-09b8-493d-8e66-289665d672f5/edit?invitationId=inv_c62c0815-6391-4479-9516-5481af4c2ec8&page=0_0#, 2022. Accessed: 2023-4-12.

[2] M. Alanazi, A. Alanazi, and A. Khodaei. Long-term solar generation forecasting. pages 1–5. IEEE/PES Transmission and Distribution Conference and Exposition (T&D), 2016.

[3] International Renewables Agency. Renewables 2021. https://www.iea.org/reports/renewables-2021, 2021. Accessed: 2022-12-15.

[4] Gilbert M. Masters. *Renewable and Efficient Electric Power Systems*. Wiley, New York, 2013.

[5] Dzung D. Nguyen, Brad Lehman, and Sagar Kamarthi. Solar photovoltaic array's shadow evaluation using neural network with on-site measurement. 10, pages 44–49. IEEE Transaction on Energy Conversion, 2007.

[6] S. J. Wellby and N. A. Engerer. Categorizing the meteorological origins of critical ramp events in collective photovoltaic array output. *Journal of Applied Meteorology and Climatology*, 55(16):1323–1344, 2016.

[7] N. A. Engerer. Simulating photovoltaic array performance using radiation observations from the oklahoma mesonet. Master's thesis, University of Oklahoma, Norman, OK, 2011.

[8] Unruh Jewel. Limits on cloud-induced fluctuation in photovoltaic generation. 5, pages 8–14. IEEE Transmission of Energy Conversion, 1990.

[9] Muhammad H. Rashid. *Power Electronics Handbook (Second Edition)*. Elsevier, Cambridge, MA, 2007.

[10] G. M. Tina, C. Ventura, S. Ferlito, and S. De Vito. A state-of-art-review on machine-learning based methods for pv. *Applied Sciences*, 11(7550):1–10, 2021.

[11] S. Theocharides, M. Theristis, G. Makrides, M. Kynigos, C. Spanias, and G.E. Georghiou. Comparative analysis of machine learning models for day-ahead photovoltaic power production forecasting. *Energies*, 14(1081):9–10, 2021.

[12] W. Glassley, J. Kleissl, C.P. Dam, H. Shiu, and J. Huang. California renewable energy forecasting, resource data and mapping executive summary. *Public Interest Energy Research Program*, pages 1–135, 2010.

[13] A. Yona, T. Senjyu, A.Y. Saber, T. Funabashi, H. Sekine, and C.H. Kim. Application of neural network to 24-hour-ahead generating power forecasting for pv system. pages 1–6. 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2007.

[14] Dávid Markovics and Martin Mayer. Comparison of machine learning methods for photovoltaic power forecasting based on numerical weather prediction. *Renewable and Sustainable Energy Reviews*, 151:1–15, 2022.

[15] T. Zhu, Y. Guo, Z. Li, and C. Wang. Solar radiation prediction based on convolution neural network and long short-term memory. *Energies*, 14(8498):1–9, 2021.

[16] Manajit Sengupta, Yu Xie, Anthony Lopez, Aron Habte, Galen Maclaurin, and James Shelby. The national solar radiation data base (nsrdb). *Renewable and Sustainable Energy Reviews*, 89:51–60, 2018.

[17] Special Interest Groups Energy. Energy systems and informatics. https://energy.acm.org/resources/, 2023. Accessed: 2023-03-20.

[18] NREL. Nsrdb: National solar radiation database. https://nsrdb.nrel.gov/data-viewer/, 2023. Accessed: 2023-03-20.

[19] UMASS Amherst Computer Science. Cumulative inverter performance. http://s38210.mini.alsoenergy.com/Dashboard//2a5669735064477642415 54b772b71593d, 2023. Accessed: 2023-03-20.

[20] California DGStats. Csi 15-minute interval data. https://www.californiadgstats.ca.gov/downloads/, 2023. Accessed: 2023-03-20.

[21] Shaurya Singh. Solar irradiance concepts: Dni, dhi, ghi & gti. https://www.yellowhaze.in/solar-irradiance/, 2022. Accessed: 2022-12-15.

[22] National Oceanic and Atmospheric Administration. Dew point vs. humidity. https://www.weather.gov/arx/why_dewpoint_vs_humidity, 2023. Accessed: 2023-03-27.

[23] DBpedia. Solar zenith angle. https://dbpedia.org/page/Solar_zenith_angle, 2022. Accessed: 2022-12-15.

[24] Copernicus Global Land Service. Surface albedo. https://land.copernicus.eu/global/products/sa, 2020. Accessed: 2022-12-15.

[25] Abdullahi Abubakar Mas'ud. Comparison of three machine learning models for the prediction of hourly pv output power in saudi arabia. *Ain Shams Engineeringh Journal*, 13:2–5, 2022.

[26] Yale. Linear regression. http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm, 1998. Accessed: 2023-03-27.

[27] Angela Shi. Sgdregressor with scikit-learn: Untaught lessons you need to know. https://towardsdatascience.com/sgdregressor-with-scikit-learn-untaught-lessons-you-need-to-know-cf243043 ~:text=The%20SGDRegressor%20algorithm%20uses%20stochastic,with%20respect%20to%20the%20parameters, 2023. Accessed: 2023-03-27.

[28] Lumivero. Partial least squares. https://www.xlstat.com/en/solutions/features/partial-least-squares-regression#:~:text=The%20Partial%20Least%20Squares%20regression,used%20to%20perfom%20a%20regression.&text=Some%20programs%20differentiate%20PLS%201,is%20only%20one%20dependent%20variable., 2023. Accessed: 2023-03-27.

[29] Jim Frost. Multicollinearity in regression analysis: Problems, detection, and solutions. https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/, 2020. Accessed: 2023-4-6.

[30] Pennsylvania State University. Ridge regression. https://online.stat.psu.edu/stat857/node/155/, 2018. Accessed: 2023-03-27.

[31] James A. Godwin. Ridge, lasso, and elastic net regression. https://towardsdatascience.com/ridge-lasso-and-elasticnet-regression-b1f9c00ea3a3, 2021. Accessed: 2023-03-27.

[32] MathWorks. Lasso and elastic net. https://www.mathworks.com/help/stats/lasso-and-elastic-net.html, 2022. Accessed: 2023-4-2.

[33] Scikit learn Developers. Scikit learn kernel ridge. https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html#:~:text=Kernel%20ridge%20regression%20(KRR)%20combines,function%20in%20the%20original%20space., 2023. Accessed: 2023-03-27.

[34] Christopher M. Bishop. *SVMs for Regression*, pages 338–345. Springer, New York, NY, 2006.

[35] Scikit learn Developers. Nusvr. https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html, 2023. Accessed: 2023-03-27.

[36] Debottam Mukherjee, Samrat Chakraborty, Pabitra Kumar Gucchait, and Joydeep Bhunia. Machine learning based solar power generation forecasting with and without an mppt controller. pages 44–46, New Jersey, NJ, 2020. IEEE Xplore.

[37] Ashwin Raj. A quick and dirty guide to random forest. https://towardsdatascience.com/a-quick-and-dirty-guide-to-random-forest-regression-52ca0af157f8, 2020. Accessed: 2023-03-27.

[38] University of Texas. Sampling with replacement and sampling without replacement. https://web.ma.utexas.edu/users/parker/sampling/repl.htm, 2023. Accessed: 2023-04-06.

[39] Simeon Kostadinov. Understanding gru networks. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be, 2017. Accessed: 2023-03-27.

[40] Aland Gupta. Grated recurrent unit networks. https://www.geeksforgeeks.org/gated-recurrent-unit-networks/, 2023. Accessed: 2023-03-27.

[41] Scikit learn Developers. Neural network models (supervised). https://towardsdatascience.com/ridge-lasso-and-elasticnet-regression-b1f9c00ea3a3, 2023. Accessed: 2023-03-27.

[42] Chun-hung Liu, Jyh-Cherng Gu, and Ming-Ta Yang. A simplified lstm neural networks for one day-ahead solar power forecasting. *IEEE Access*, 9:17175–17178, 2021.

[43] Keras. Adadelta. https://keras.io/api/optimizers/adadelta/, 2012. Accessed: 2023-03-27.

[44] Cort J. Willmott, Steven G. Ackleson, Robert E. Davis, Johannes J. Feddema, Katherine M. Klink, David R. Legates, James O'Donnell, and Clinton M. Rowe. Statistics for the evaluation and comparison of models. *Energies*, 90(C5):1–8, 1985.

[45] Sibarama Panigrahi and H. S. Behera. Effect of normalization techniques on univariate time series forecasting using evolutionary higher order neural network. *International Journal of Engineering and Advanced Technology*, 3(2):1–6, 2013.

[46] Desmos. Why am i seeing a negative r^2 value? https://help.desmos.com/hc/en-us/articles/202529139-Why-am-I-seeing-a-negative-R-2-value-#:~:text=In%20practice%2C%20R2%20will,the%20mean%20of%20the%20data., 2020. Accessed: 2023-4-6.