Wondering what to blame? Turn PV performance assessments into maintenance action items through the deployment of learning algorithms embedded in a Raspberry Pi device.

C. Birk Jones*, Manel Martínez-Ramón†,
Craig Carmignani*, Joshua S. Stein*, and Bruce H. King*

*Sandia National Laboratories Solar PV & Grid Integration, Albuquerque, NM, U.S.A
†Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, U.S.A
Universidad Carlos III de Madrid, Leganés, Madrid, Spain

***Abstract*—Monitoring of photovoltaic (PV) systems can maintain efficient operations. However, extensive monitoring of large quantities of data can be a cumbersome process. The present work introduces a simple, inexpensive, yet effective data monitoring strategy for detecting faults and determining lost revenues automatically. This was achieved through the deployment of Raspberry Pi (RPI) device at a PV system's combiner box. The RPI was programmed to collect PV data through Modbus communications, and store the data locally in a MySQL database. Then, using a Gaussian Process Regression algorithm the RPI device was able to accurately estimate string level current, voltage, and power values. The device could also detect system faults using a Support Vector Novelty Detection algorithm. Finally, the RPI was programmed to output the potential lost revenue caused by the abnormal condition. The system analytics information was then displayed on a user interface. The interface could be accessed by operations personal to direct maintenance activity so that critical issues can be solved quickly.**

***Index Terms*—Gaussian Process, support vector machine, Raspberry Pi, modbus, photovoltaics, data collection, fault detection**

I. Introduction

Solar photovoltaic (PV) arrays require minimal maintenance and operations to produce electricity, because fixed tilt arrays have minimal moving parts. This differs from gas generators or a wind turbines, which have components that must be oiled, repaired, or replaced on a regular basis. Therefore, common practice for many systems has been to install the PV panels and then walk away without further monitoring. However, faults do occur and can be costly. The fault conditions pose significant financial risk because the long-term financing costs have become a significant portion of the overall cost. The financing agreements demand string performance reviews so that financial returns match expectations. Therefore, large scale developments require intensive monitoring and oversight.

Current recommendations for PV system oversight suggest metrics such as performance ratio (PR) [1], temperature corrected PR, Energy Performance Index (EPI) SAM model, EPI Regression model, and Power Performance Index (PPI). However, these approaches do not provide an effective means to detect and quantify the impact of fault conditions. Instead, the present work proposes the implementation of a low cost intelligent Raspberry Pi (RPI) device to perform advanced monitoring of PV sub-systems. The device was designed to be deployed within combiner boxes or at inverters to collect PV sensor data through Modbus communications. Then, the PV sub-system data can be analyzed by the onboard analytics that includes advanced machine learning algorithms.

The integration of an intelligent RPI device into a PV system can provide detailed assessments of string level performance that can lead to improved operations. The RPI has the capability to estimate system performance using a Gaussian Process Regression (GPR) algorithm. It also has a Support Vector Novelty Detection (SVND) algorithm that automatically detects faults. When a fault is detected the RPI uses the GPR estimate to compute the lost energy production caused by the fault condition. This paper describes the basic set-up of the RPI device on an actual system.

The set-up includes custom Modbus code written in Python programming language, a MySQL database, and a web-based visualization as described in Section II-A. The embedded algorithms used to estimate performance and detect faults are described in Sections II-B and II-C. The RPI analytics is running in real-time with an actual system and results from the initial 30 day period are described in Section III. The results section describes the user interface (III-A), the PV performance estimate results (III-B), fault detection accuracies (III-C), and a lost energy production calculation example (III-D).

Fig. 1. The present work performed tests on a 10.8kW$_e$ array. The array has four strings that each have 10 modules. The strings are combined prior to entering the inverter. In addition, each of the string's current and voltage are monitored.

II. Methodology

The present work integrated an intelligent, inexpensive, and deployable RPI device into a PV system. The device was programmed to monitor operations, detect faults, and estimate power losses caused by system faults automatically. The proposed device was connected directly to an analog input unit through Modbus RS-485 and TCP/IP protocols. The collected PV sensor data was stored in a local MySQL database. Then, advanced learning algorithms, that include the GPR and SVND, analyzed the data and provide feedback through a graphics-based web interface.

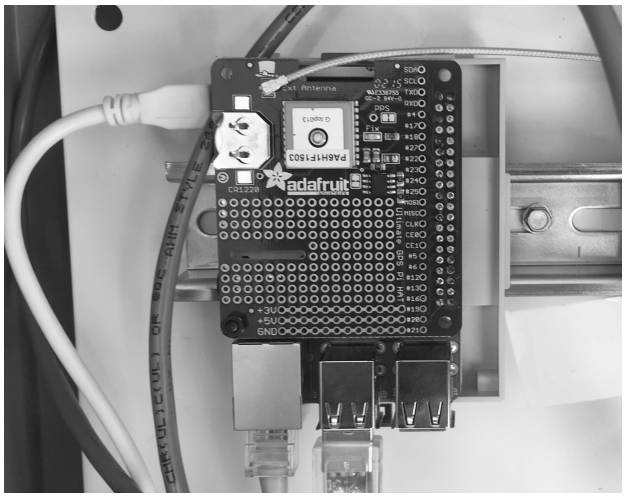

Fig. 2. The Raspberry Pi was located inside an enclosure attached to the array. It was powered by a 5V power supply, and connected to the existing data collection devices through Modbus TCP/IP and Serial ports. It also has its own GPS time clock to maintain correct date and time throughout the data collection process.

A. Device Set-up

The intelligent RPI was deployed on an actual 10.8kW$_e$ PV array (Figure 1) located in Albuquerque, New Mexico. The array had four strings of 10 modules that were combined into one prior to entering the inverter. The RPI device, shown in Figure 2, was deployed inside the combiner box and offered an energy efficient way to perform computational tasks [2]. A critical task was collecting actual sensor data. This was achieved through Modbus communications as shown in Figure 3. The device connected with the Modbus TCP/IP points available on a network created by the Gantner Q.Station 101 data collection device. A Modbus serial connection was established through the RPI USB port, and provided access to inverter data. The data was extracted from the two Modbus protocols using Python programing language script and stored in a MySQL database on the RPI.

Machine learning algorithms, that include GPR and SVND, were used to estimate actual performance and evaluate the status of a PV system. The algorithms accessed the weather and PV sensor data stored in the MySQL database. The GPR estimator used the weather data to estimate ideal current, voltage, and power values. At the same time, the SVND analyzed the stored data to define normal or abnormal conditions. The results were then inserted back into the MySQL database and were accessible for viewing through a web interface.

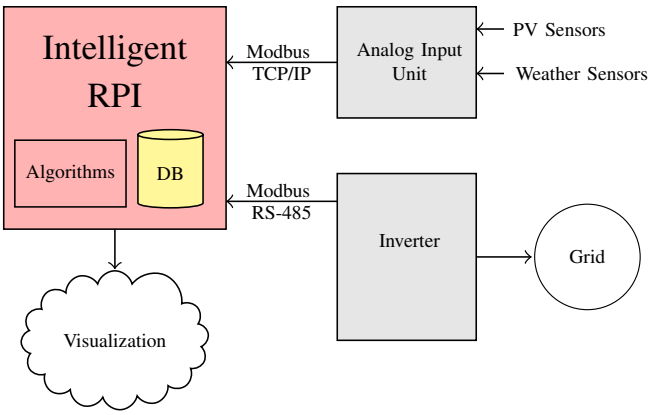

Fig. 3. The intelligent RPI had complete interoperability with the existing analog input unit and the PV system inverter. Data was collected through Modbus TCP/IP and RS-485 and stored in a local MySQL database. The machine learning algorithms analyzed the data. The data collection and analysis results were displayed through a web-based graphical interface.

B. Estimate PV Performance

The estimated PV performance could be calculated using a component-based or empirical model. For example, the Python version of PV_LIB [3] could be run on a RPI device. However, in this experiment the estimator was the GPR algorithm. The algorithm was presented with a training data set, $D = ((x_i, y)|i = 1, ...n)$. The inputs x_i included ambient temperature and solar irradiance. Its outputs y_i were current, voltage, and power.

GPR can be defined as a random process where any finite subset of these process have a joint Gaussian distribution [4]. GPR applies a distribution over functions that are specified by a mean function and a covariance function as shown in Eqn. 1.

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

The mean function, $\mu(x)$, is usually defined to be zero and the covariance $k(x, x')$ defines the prior properties of the functions considered for inference [5]. The k in the covariance represents the kernel function which projects the data into a higher dimensional feature space to increase the computational power of the algorithm [6].

The transformation from input into the higher dimensional feature space, known as Hilbert space, was accomplished through a nonlinear transformation ($\phi(\cdot)$) shown in Eqn 2.

$$y_n = \mathbf{w}^\top \phi(\mathbf{x}_n) + \epsilon_n \tag{2}$$

This means that a nonlinear relationship has been established between the input and output observations. Since the nonlinear transformation $\phi(\cdot)$ maps the input into a Kernel Reproducing Hilbert Space (RKHS), by virtue of the Mercer's theorem [7], there exists a dot product

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \mathbf{\Sigma} \phi(\mathbf{x}') \tag{3}$$

between the transformed observations. The dot product has covariance properties where $\mathbf{\Sigma}$ is a positive semidefinite matrix.

The GP approach assumes that the error (ε) has been drawn from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. The weight vector is also modeled as a Gaussian random variable with a prior that considered $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w)$. Finally, it is assumed that ϵ_n is an independent and identically distributed process. It consists of an Additive White Gaussian Noise (AWGN), and is independent from the observations ($\mathbf{x}_n$) as well as the parameters described by $\mathbf{\Sigma}_w$. In this case, the covariance $\mathbb{E}(y_n y_m|\mathbf{X})$ of regressors is

$$\begin{aligned}\mathbb{E}(y_n y_m|\mathbf{X}) &= \mathbb{E}\left((\mathbf{w}^\top\phi(\mathbf{x}_n)+\epsilon_n)(\mathbf{w}^\top\phi(\mathbf{x}_m)+\epsilon_m)\right)\\ &= \phi^\top(\mathbf{x}_n)\mathbb{E}(\mathbf{w}\mathbf{w}^\top)\phi(\mathbf{x}_m)+\mathbb{E}(\epsilon_m\epsilon_m)\\ &= \phi^\top(\mathbf{x}_n)\mathbf{\Sigma}_w\phi(\mathbf{x}_m)+\mathbb{E}(\epsilon_m\epsilon_m)\\ &= k(\mathbf{x}_n,\mathbf{x}_m)+\sigma^2\delta(m-n)\end{aligned} \tag{4}$$

where $\mathbf{X}$ is a column matrix containing all input observations $\mathbf{x}_n$. The noise (ϵ_n) is an AWGN process independent of $\mathbf{x}_n$. The covariance matrix, $\mathbf{\Sigma}_w$, is assumed to be random process and $\phi^\top(\mathbf{x}_n)\mathbf{\Sigma}_w\phi(\mathbf{x}_m)$ is a dot product in the Hilbert space. Therefore, the last expression of Eqn. (4) can be considered a kernel expression of this dot product and the covariance matrix of the regressor sequence y_n can be written as

$$\mathbf{K}_{y,y} = \mathbf{K} + \sigma\mathbf{I} \tag{5}$$

where $\mathbf{K}$ is the matrix of kernel dot products between observations. This kernel matrix of the observations during noise fee conditions is equivalent to the covariance matrix of the regressors. Then, if the values $\mathbf{x}_n,\ y_n$ are used as the training samples and a new sample $\mathbf{x}_*$ is added, the prediction over this sample is $\mathbf{f}_* = \mathbf{w}^\top\phi(\mathbf{x}_*)$.

The joint probability distribution of the process has a zero mean Gaussian whose covariance matrix contains the covariance of the new estimation, one of the training regressors, and the cross covariance between them.

$$p(\mathbf{f}_*, \mathbf{y}) = \mathcal{N}\left(\mathbf{0},\ \begin{matrix}\mathbf{K}_{*,*} & \mathbf{K}_{y,*}\\ \mathbf{K}_{*,y} & \mathbf{K}_{y,y}\end{matrix}\right) \tag{6}$$

$\mathbf{K}_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$ and $\mathbf{K}_{*,y} = \mathbf{K}_{y,*}^\top$ is the row vector of all dot products $k(\mathbf{x}_*, \mathbf{x}_n)$. The GPR goal is to compute the *predictive* posterior over the new sample $\mathbf{x}_*$ given the training data. Using the Bayes' rule, this posterior has the form of another Gaussian, with mean and variance given by

$$\begin{aligned}\boldsymbol{\mu}_* &= \mathbf{K}_{y,*}\mathbf{K}_{y,y}^{-1}\mathbf{y}\\ \boldsymbol{\sigma}_*^2 &= \mathbf{K}_{*,*} - \mathbf{K}_{y,*}^\top\mathbf{K}_{y,y}^{-1}\mathbf{K}_{y,*}\end{aligned} \tag{7}$$

In absence of noise, the mean matches the minimum mean square error prediction and the arbitrary value of σ^2 matches the Kernel Ridge Regression prediction [6]. The advantage of the GPR is that it offers a posterior distribution over the prediction rather than a prediction alone. Its variance is reduced with respect to the variance of its prior in a quantity $\mathbf{K}_{y,*}^\top\mathbf{K}_{y,y}^{-1}\mathbf{K}_{y,*}$.

The noise variance, σ^2, and the kernel parameters are, in principle, free parameters and they must be adjusted. The procedure starts with the computation of the log-likelihood of the regressors y_n. Then, this likelihood can be maximized taking derivatives over the parameters and applying a standard gradient ascent. The log likelihood is not necessarily convex, so different initializations may be necessary to achieve a good convergence. Finally, non Gaussian distributions for the likelihood can be assumed, but then the GPR does not have closed solutions. Nevertheless, in many cases the GPR algorithm [8] can be applied to obtain asymptotically optimal solutions.

C. Fault Detection

The detection of faults was performed using a Support Vector Machine (SVM) algorithm. SVM algorithms can learn through supervised [9], [10] or unsupervised [11], [12] methods. It learns by separating different classes in a training data set with an optimal hyperplane. The hyperplane is created by maximizing the minimum distance to the training points closest to the plane. This is accomplished by mapping the input vectors into a high dimension feature space. In this space, a surface is constructed that separates the data.

For this experiment, the SVND algorithm was used to detect faults. The SVND is an unsupervised algorithm that learns a decision function for novelty detection. In this case, the algorithm performed a fit on an input array only and it did not consider class labels. The SVND algorithm used a radial basis kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x}-\mathbf{x}'||^2) \tag{8}$$

with a margin parameter, μ, equal to 0.12 and a γ equal to 45.

The SVND algorithm discovered a hyperplane that separated the data with a maximum margin in the feature space through the cost function

$$min\{\frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho\} \tag{9}$$

that is constrained to

$$\begin{aligned}(\mathbf{w}\cdot\phi(x_i)) &\geq p - \xi_i\\ \xi_i &\geq 0\end{aligned} \tag{10}$$

The **w** and b were the gradient and intercept of the linear decision boundary in the feature space, ξ_i were positive "slack variables", and ρ was the bias term. The μ parameter set the upper bound on the fraction of outliers and a lower bound on the number of training examples used as a support vector. The decision function then became

$$f(\mathbf{x}_*) = sign(\mathbf{w}\cdot\phi(x_i) - \rho) \tag{11}$$

using the Lagrange techniques and a kernel function for the dot-product calculation.

D. Experiment

The embedded GPR and SVND algorithms were run in real-time with the actual system and their respective results were displayed on a web interface. This paper provides an overview of the systems performance by focusing on a 31 day period between January 28 and February 28, 2016. The experiment began the training process on January 28th and the first estimates and faults detection algorithms were conducted on data from January 29th. The on-line training approach updated the overall training set as time went on and as more data became available.

III. Results

The results for this integration and analysis experiment were broken out into three sections. The first section (Sec. III-A) described the user interface that the RPI provides through an ethernet or internet connection. The next section, Sec. III-B, reviews the results form the GPR estimation. The third section (Sec. III-C) describes the fault detection results. The final results section (Sec. III-D) describes an example calculation that highlighted the approximation of lost power caused by a fault condition.

A. Web-based Visualization

The web-based user interface includes a live data stream of actual voltage, current, and power data as shown in Figure 4. The graphs included in the web-site plot both the actual and estimated values with respect to time. The site also graphs ac-

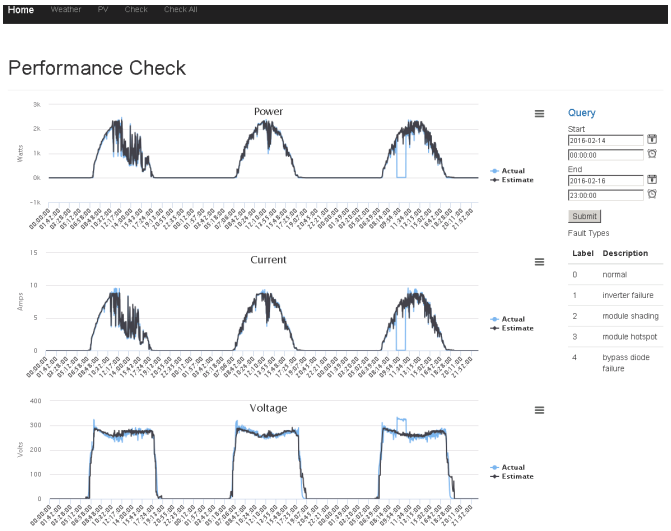

Fig. 4. The web-based visualization allows the user to review current and past data weather and performance data.

tual weather data. This includes plane of array solar irradiance, ambient temperature, and module temperature. Additionally, the web-site allows the user to query current and past data through simple dropdown menus.

B. PV Performance Estimates

The GPR algorithm estimated DC current, voltage, and power for the PV system. The estimation could be performed in real-time or at the end of each day depending on the operations needs. In this case, the RPI was set-up to estimate performance every hour throughout the entire day. On average,

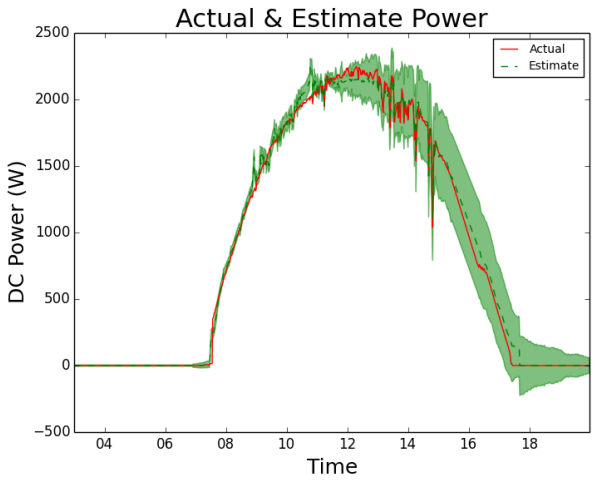

Fig. 5. Gaussian Process estimate for DC power has significant uncertainty at the beginning of the on-line learning process.

the computational time was about 2-4 minutes on the RPI to train and then estimate a single hour. The training process was conducted in an on-line learning approach. This meant that the process began the day after the data collection started. The algorithm was trained on the previous day and then estimated the current day's performance. Then, the next day estimates

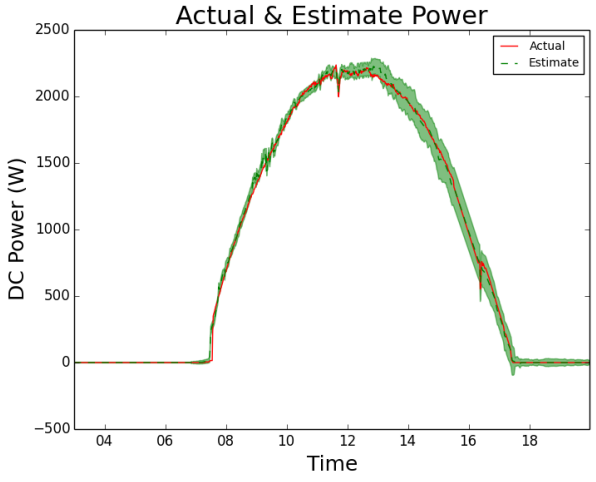

Fig. 6. The Gaussian Process estimate of DC power is more confident after a few days of training in the on-line learning approach.

used the previous two days for training and the most current data for testing. This methodology continued, which meant that as time went on the training set expanded.

It was evident that as the training increased the uncertainty of the GPR results improved. For example, Figure 5 shows the

GPR estimation for power over a single day. The estimated has a significant estimated range in the afternoon caused by the uncertainty associated with the training data set. However, after a few days of training the uncertainty decreased, and the GPR estimation had a smaller distribution as shown in Figure 6.

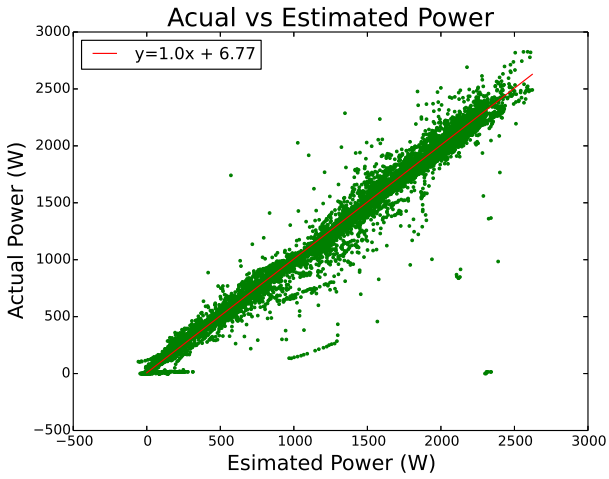

Fig. 7. The Gaussian Process estimates had a strong linear relationship with the actual values. The slope of the linear fit line was 1 and the intercept was calculated to be 6.77.

This experiment, which was conducted over a 30 day period, produced estimated power results that had a strong linear relationship to the actual data as shown in Figure 7. The linear fit had a slope of 1 and an intercept equal to 6.77. The linear relationship between the actual and estimated values included actual fault conditions. This inclusion may be the reason for the numerous outlier data points plotted in Figure 7.

C. Fault Detection

The fault detection algorithm embedded in the RPI was the SVND described in Section II-C. The algorithm accurately detected normal and abnormal behavior over the 30 day test period. The abnormal behavior included module shading, inverter failure, and module hot spots. The algorithm used voltage, current, power, solar irradiance, and module temperatures as the inputs. The outputs were either 0 for normal or 1 for a fault condition. The fault detection results were analyzed based on receiver operating characteristic (ROC) curve shown in Figure 8.

The ROC curve is a graphical plot that describes binary classification performance. The ROC plot, shown in Figure 8, described the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The desired ROC curve should have an area under the curve that is greater than 0.5. An area that is less than or equal to 0.5 would indicate that the algorithm has a bad predictor and has produced worse results than what a 50/50 guess would provide. In this experiment, the SVND predicted was well above 0.5 at 0.98 as shown in Figure 8. Based on the ROC curve result the optimal threshold was determined to be -12. This threshold value was then permanently set in the SVND that was embedded into the RPI device.

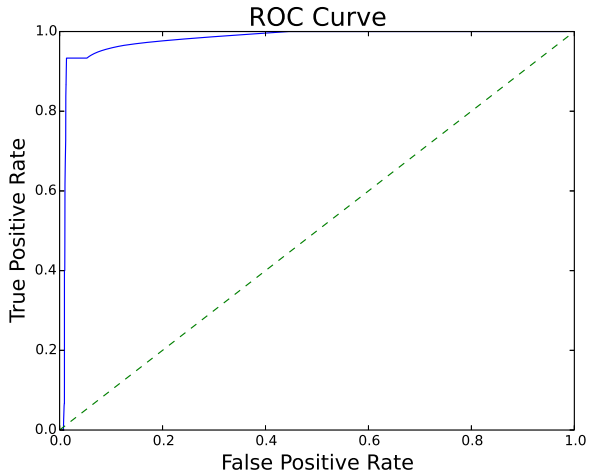

Fig. 8. The receiver operator curve produced by the SVND algorithm for the 30 day test. The curve produced a area much higher than 0.5 at 0.98.

The optimal threshold of -12 was based on the algorithms ability to provide both a high TPR and a low FPR rate. After setting the threshold, the SVND algorithm was applied to the 20 days of previously unseen data. The algorithm was able to detect abnormal conditions at a very high accuracy as shown in Figure 9. Figure 9 provides a two dimensional visualization of how well the algorithm differentiated normal and abnormal behavior. In this example, the data points arranged in a diagonal line on the power versus irradiance plot were normal performance data. Along this diagonal there no fault were identified and therefore no large, colored circles surrounding any of the points. Whereas, the data not found on the diagonal

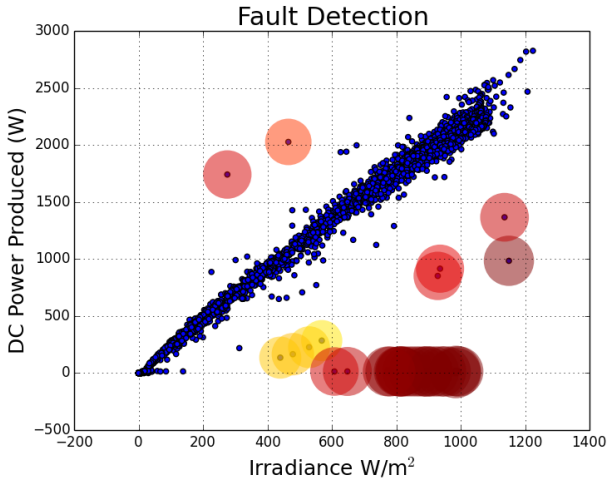

Fig. 9. The SVND results for the 30 day period are depicted in this 2 dimensional plot of DC power against irradiance. The data points plotted within the diagonal line were considered normal and the others highlighted by the large circles were abnormal fault conditions.

are highlighted by the different circles. The range of colors indicate the respective scores associated with the data point

assigned by the SVND. The scores can be considered an estimate of probability that the given data point is a fault. The data points surrounded by the larger circles indicate that the point has a higher probability of being a fault. Whereas the points with the smaller circles have lower probability.

D. Lost Production

The final task, performed by the intelligent RPI, was to estimate the lost energy production caused by the fault conditions. For example, Figure 10 compares the power output for a single string in the array. Over the course of the day the string experienced shading that decreased the overall power output. The GPR estimate had an overall energy of 12.39kWh and the actual system was calculated to produced 11.68kWh. Therefore, the actual system produced 0.71kWh less than

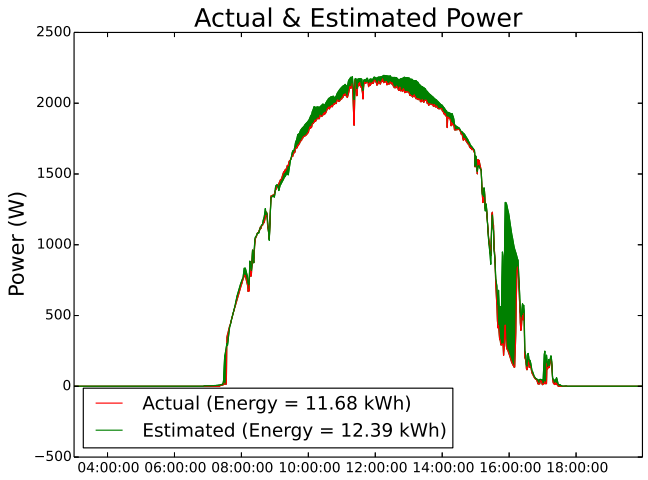

Fig. 10. The difference between the actual and estimated was used to calculate the lost energy. In this case a total of 22.9kWh were not produced.

expected. This decrease in power equated to 6% loss in energy production. This may not seem like much, but if an issue like this went undetected in a large one megawatt system that produced 1.7 million kWh a year it would amount to around 102,000 kWh in lost energy over an entire year.

IV. Conclusion

The intelligent RPI device was integrated into an existing PV array. It successfully communicated with the inverter and analogy sensor device through Modbus serial and TCP/IP communications using Python scripts. The PV performance and weather data was stored in a local MySQL database. The database was then accessed by GPR and SVND algorithms. The GPR algorithm, embedded in the RPI, was trained in an on-line learning approach and was able to accurately estimate PV performance. The GPR results had a very strong linear relationship with the actual data. The detection of faults within the PV array was performed using a support vector machine novelty detection algorithm. This algorithm was evaluated using the ROC curve and then applied to the data set to identify fault conditions. The ROC curve results showed that it could detect faults at a high TPR while maintaining a low FPR. The RPI device also calculated the lost electrical production caused by the fault condition. All of these results and the actual data were stored in the local database. A web-based interface accessed the data and displayed the actual and GPR estimates graphically. The visualization also provided a table of the detected faults and it's associated time-stamp.

The RPI device is currently still a research tool and has not been released for commercial use. Continued developments and testing are required to improve it's ability to adapt to various weather conditions as well as different PV array sizes and types.

Acknowledgment

This work was supported by the U.S. Department of Energy SunShot Initiative

The work was also partly supported by Spanish Ministerial Commission of Science and Technology (TEC2014-52289-R) and Comunidad Autónoma de Madrid (PRICAM P2013ICE-2933) grants.

Sandia National Laboratories is a multi-program managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] V. Sharma, A. Kumar, O. S. Sastry, and S. S. Chandel, "Performance assessment of different solar photovoltaic technologies under similar outdoor conditions," *Energy*, vol. 58, pp. 511–518, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544213004994

[2] E. Upton and G. Halfacree, *Raspberry Pi User Guide*. John Wiley & Sons, Sep. 2014.

[3] R. W. Andrews, J. S. Stein, C. Hansen, and D. Riley, "Introduction to the open source PV LIB for python Photovoltaic system modelling package," in *2014 IEEE 40th Photovoltaic Specialist Conference (PVSC)*, Jun. 2014, pp. 0170–0174.

[4] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, Aug. 2012.

[5] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[6] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, Jun. 2004, published: Hardcover.

[7] M. A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[8] T. Minka, "Expectation propagation for approximate Bayesian inference," in *Proc. of 17th Conference in Uncertainty in Artificial Intelligence*. Seattle, Washington, USA: University of Washington, 2001, pp. 362–369.

[9] A. J. Smola and B. Schlkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[10] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 1–32, 1998.

[11] D. Tax and R. P. W. Duin, "Support vector domain description," *Pattern Recognition Letters*, vol. 20, pp. 1191–1199, 1999.

[12] B. Schlkopf, R. C. Williamson, A. Smola, and J. Shawe-Taylor, "Support Vector Method for Novelty Detection," in *Advances in Neural Information Processing Systems 12*, Denver, CO, 1999.