Scientific Analysis and Modeling of Effect of Various Climatic Conditions on the Performance of Photovoltaic Modules

UNDERGRADUATE THESIS

Submitted in partial fulfillment of the requirements of
BITS F422T, Thesis

by

Ananya Kshatrapati Shivaditya
2013B1A70792G

Under the supervision of

Dr. Mridul Sakhuja
Solar Energy Research Institute of Singapore, National University of Singapore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
K. K. BIRLA GOA CAMPUS

May 2018

Acknowledgements

Thank you Dr. Mridul, for your continuous support.

Thank you Meenal ma'am, for your guidance.

Thank you Dr. Franco and Parvathy.

Sincere gratitude to my university, Birla Institute of Technology and Science, K. K. Birla Goa Campus, for making this thesis possible.

To the professors, staff members and students at Solar Energy Research Institute of Singapore, thank you all for your assistance and well wishes.

Certificate

This is to certify that the thesis entitled, "*Scientific Analysis and Modeling of Effect of Various Climatic Conditions on the Performance of Photovoltaic Modules*" and submitted by Ananya Kshatrapati Shivaditya ID No. 2013B1A70792G in partial fulfillment of the requirements of BITS F422T Thesis embodies the work done by her under my supervision.

Supervisor
Dr. Mridul Sakhuja
Research Fellow Project Manager,
TruePower Alliance,
Solar Energy Research Institute of
Singapore, National University of Singapore
Date:

Abstract

Scientific Analysis and Modeling of Effect of Various Climatic Conditions on the Performance of Photovoltaic Modules

Under the Supervision of Dr. Mridul Sakhuja
Under the Co-Supervision of Dr. Meenal Kowshik

Second Semester 2017-2018
Ananya Kshatrapati Shivaditya, 2013B1A70792G

Birla Institute of Technology and Science, Pilani

This report is a part of the TruePower Project, which is aimed at building advanced indoor test facilities for PV modules in all relevant environmental conditions combined with high precision outdoor module plus system level testing to explain real world impact of climatic factors on PV system performance. Using machine learning, we predict performance of a PV system in terms of Open-Circuit Voltage(V_{oc}), Short-Circuit Current(I_{sc}) and Power at Maximal Power Point(P_{mpp}) from the module's characteristics along with the weather conditions at that time. After achieving near perfect accuracy or prediction, these models are analyzed to understand how the input features affect the quantity to be predicted, and to what extent. With this information, a set of climate conditions are derived, under which performance of a module is near best, across all the different module technologies. This proposed set of testing conditions will help estimate best case performance of a module in it's installed location.

Contents

Chapter 1

Introduction

1.1 Overview of the TruePower Project

The TruePower Project is aimed at building advanced indoor test facilities for Photovoltaic modules in all relevant environmental conditions combined with high precision outdoor module plus system level testing to explain real world impact of climatic factors on PV system performance. All relevant climate types, i.e. Tropical, Temperate and Arid are used for data collection. This data is analyzed and a set of indoor test conditions are proposed for better estimation of a module's performance in the climate type that it is installed in.

1.2 Data Collection

Data was collected from the solar and meteorological site in Australia. Prediction of 3 performance parameters, Open-Circuit Voltage (V_{oc}) measured in Volts, Short-Circuit Current (I_{sc}) measured in Amperes and Power measured in Watts at Maximal Power Point (P_{mpp}) are made from measured meteorological data, consisting of the following fields:

1. Module Temperature
2. Ambient Temperature
3. Average Photon Energy
4. Ambient Humidity
5. Inplane Irradiance from Si sensor
6. Rainfall
7. Wind Speed
8. Wind Direction

The datasets were broken up into three, on the basis of the technology of the module.

1.3 Machine Learning Models

Scikit-learn is a python library, widely used for implementation of machine learning models. It supports a fanfare of different algorithms for regression as well as classification, including Support Vector Machines, Multilayer Perceptron and Random Forest to name some of the powerful ones. In this study Random Forest Regression was found to give the highest scores of prediction, for each of the performance parameters across the different technology types of modules.

1.3.1 Random Forest Regression

Regression done by a Random Forest (RF) is a good way to minimize skewed prediction due to a few stray data points, and it also easily evades the problem of overfitting. This model was chosen for all three performance parameters' prediction, with varied number of trees in the forest, and varied maximum depth of each tree.

The RF model is a collection of decision trees, with the mean of predictions from all trees being the final prediction. Each tree is trained on a random subset of the complete dataset, and works by separating the data into smaller sized collections at each node, not necessarily of the same size but of the lowest variance within a collection. The boundary for bifurcations is the value for an attribute of the dataset which divides the data into two subsets of lowest variance. The attribute is randomly chosen at each node.

As explained, there is a high degree of randomness in the algorithm, including the training subset of each tree, and the attribute chosen at each node. This makes the model robust against overfitting.

The technique of minimizing variance at each node makes the model extremely accurate. A new prediction is made by allowing each tree to classify the given test data point to one of its leaves. The mean of the training data points which reached that leaf is taken to be that tree's prediction.

1.4 Feature Importance

The importance of each of the features was then calculated using the "mean decrease impurity" of the RF model, for each of the three performance parameters. These feature importances were examined and verified with the established theory of solar modules.

Steps to compute the feature importance values of a single tree:
1. Initialize an array feature_importances of all zeros with size equal to the number of features.
2. Traverse the tree, and for each internal node that splits on feature$_i$ you compute the error reduction of that node multiplied by the number of samples that were routed to the node and add this quantity to the importance of feature$_i$ in the array.

The error reduction depends on the impurity criterion that is used, here we use Mean Squared Error. It is the impurity of the set of examples that gets routed to the internal node minus the sum of the impurities of the two partitions created by the split.

1.5 Derivation of Arid Testing Conditions

The next step is to ascertain a set of features which allow best performance of the solar cell in the arid regions of Australia, in terms of all three of the module technologies, namely Thin-Film CdTe, Multicrystalline and CiS substrate glass. One module was created for each performance parameter V_{oc}, I_{sc} and P_{mpp}, and the aim is to find convergence of weather conditions for best performance across all the types of module technologies.

The most important feature is the basis for derivation. The radiation energy per unit area is expressed in terms of the most important feature, and the remaining features are derived from the weighted average of the most frequent occurrences of the most important feature.

Future work could also include repeating this work for cold climates in Germany.

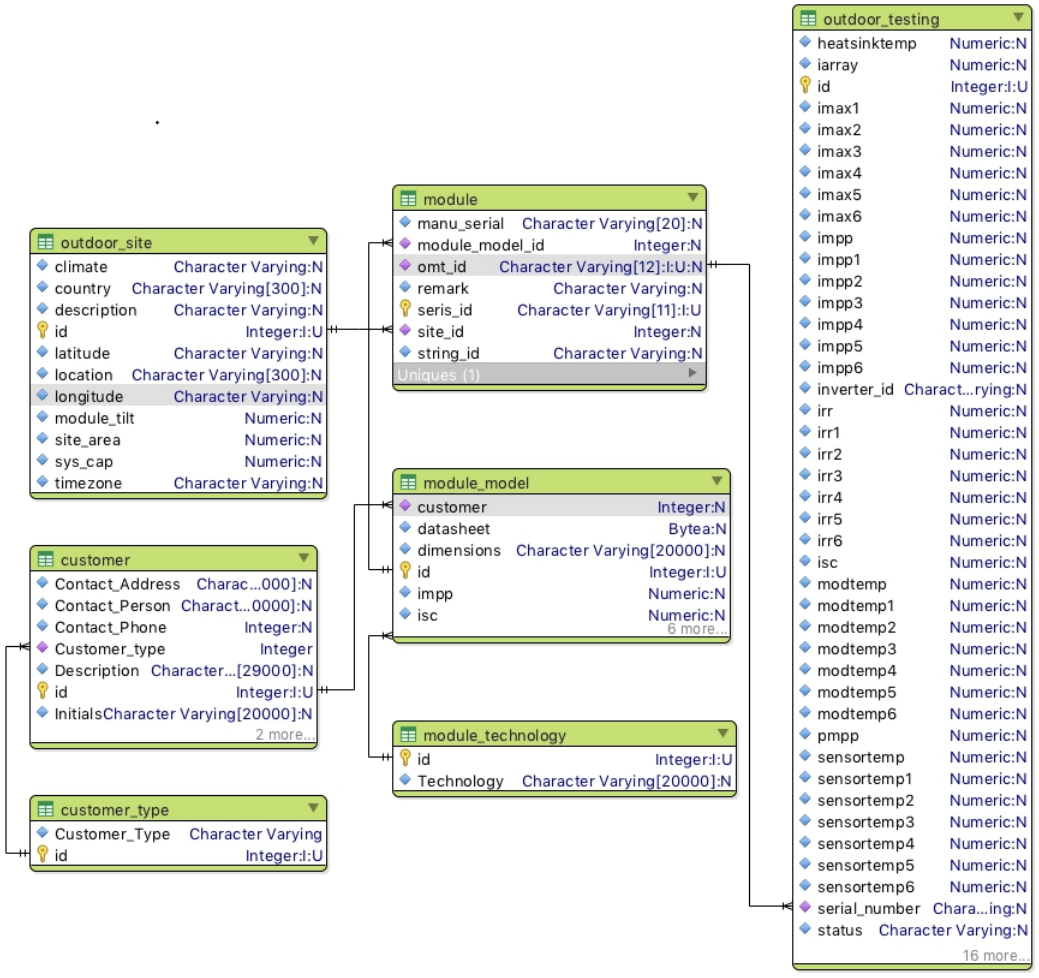
Chapter 2

Data Collection

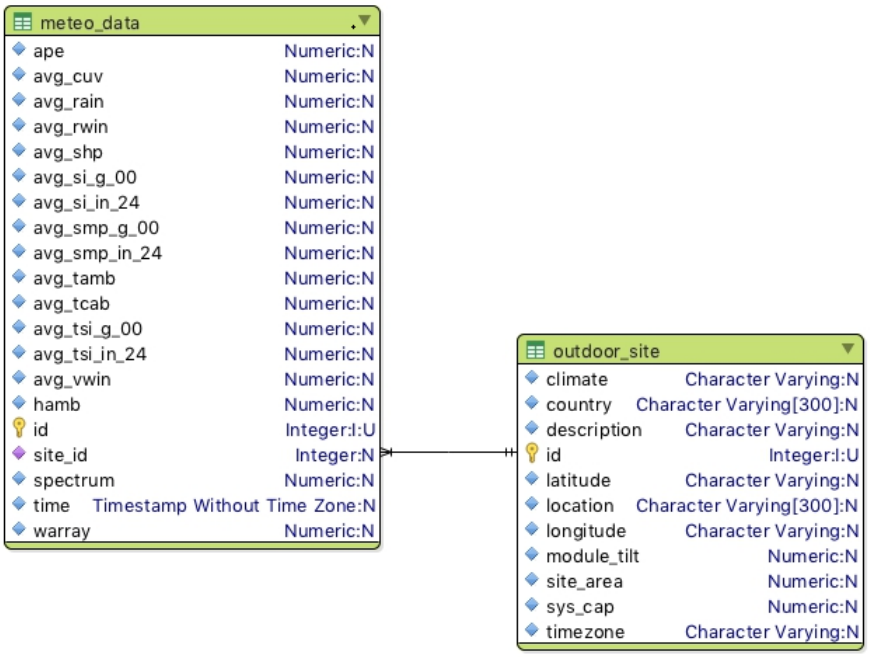
2.1 Database Schema

Data was collected from Alice Springs, in the arid regions of Australia. The solar modules were installed at a module tilt of 23.7 °. The database schema was setup as follows, with tables separated for readability:

1. Outdoor Measurement Data



2. Metorological Data



2.2 Shape of extracted Dataset

A steady flow of entries is maintained from July 2017 till the present date, approximately 9 months of data. The database was queried for measurements of V_{oc}, I_{sc} and P_{mpp}, and the recorded climatic conditions at that instant. A separate dataset was recorded for each of the three technology types of the modules. This was done to enable comparative study of each technology's dependence on environmental variables.

Size of each dataset is listed below:
1. Technology 1: Thin Film CdTe
74,27,040 rows and 8 columns

2. Technology 2: Multicrystalline
1,46,81,800 rows and 8 columns

3. Technology 3: CiS substrate glass
74,14,320 rows and 8 columns

Details of the 8 columns are given here:

1. Module Temperature (mod_temp):
Temperature (° Celcius) of the module, measured and assumed to be affected by irradance, wind, rain etc. of the atmosphere.

2. Ambient Temperature (tamb):
Ambient temperature in degrees Celcius, of the surroundings of the installed module.

3. Average Photon Energy (ape):
Aims to characterize the energetic distribution in an irradiance spectrum. It is obtained by dividing the irradiance [W/m or eV/m/sec] by the photon flux density [number of photons/m/sec].

4. Ambient Humidity (hamb):
Ambient humidity in % relative humidity, of the surroundings of the installed module.

5. Inplane Irradiance from Si sensor (inplane_irr_si):
Irradiance in Watts/meter squared along the plane of the solar module, from a second Silicon sensor.

6. Rainfall (rain):
Rainfall in mm measured at the site of installation of the module.

7. Wind Speed (wind_speed):
Speed of the wind in meters per second at the site of installation of the module.

8. Wind Direction (wind_dir):
Direction of the wind in degrees at the site of installation of the module.

2.3 Training Set and Test Set

The dataset was shuffled, and 70% of it was kept in the Training Set. The remaining 30% of the dataset was reserved to determine the model's accuracy, as Test Set.

Chapter 3

Methodology: Building the Models

3.1 Use of Scikit-Learn

Scikit-learn is a python library, widely used for implementation of machine learning models. It supports a fanfare of different algorithms for regression as well as classification, including Support Vector Machines, Multilayer Perceptron and Random Forest to name some of the powerful ones.

In machine learning, often the model used is a black-box, taking features as input and spitting out a prediction or a classification as the output. It is difficult to decipher how the prediction or classification was made, for most ML models. But for this study, we do require an understanding of how the inputs affect the prediction made.

In accordance with this aim the Random Forest Regression model from Scikit-Learn was implemented, famed to produce very high accuracies of prediction. As an added feature, the RF Regressor also includes a function for ascertaining the importance of each of the features fed into the model in making the prediction.

3.2 Random Forest Regression (RF)

Nine RF regression models were built, 3 sets, for each technology type, with a model for each of the 3 performance variables.

Hyperparameters were tuned at:
maximum depth of each tree = 20
maximum number of trees in the forest = 200

3.3 Results

The prediction scores are tabulated below:

Technology 1: Thin-Film CdTe
1. Open-Circuit Voltage (V_{oc}): 99.9133967771 %
2. Short-Circuit Current (I_{sc}): 99.9894596048 %
3. Power at Maximal Power Point (P_{mpp}): 99.9715863941 %

Technology 2: Multicrystalline
1. Open-Circuit Voltage (V_{oc}): 99.511994171 %
2. Short-Circuit Current (I_{sc}): 99.9737203396 %
3. Power at Maximal Power Point (P_{mpp}): 99.7450063617 %

Technology 3: CiS substrate glass
1. Open-Circuit Voltage (V_{oc}): 99.678031771 %
2. Short-Circuit Current (I_{sc}): 99.9696829837 %
3. Power at Maximal Power Point (P_{mpp}): 99.901236974 %

As observed, the Random Forest model provides extremely high accuracies of prediction based on the features provided. Each feature's hand in producing this accuracy is discussed below.

3.4 Feature Importances

The importance of each of the features was then calculated using the "mean decrease impurity" of the RF model, for each of the three performance parameters. These feature importances were examined and verified with the established theory of solar modules.

Steps to compute the feature importance values of a single tree:
1. Initialize an array feature_importances of all zeros with size equal to the number of features.
2. Traverse the tree, and for each internal node that splits on feature$_i$ you compute the error reduction of that node multiplied by the number of samples that were routed to the node and add this quantity to importance of feature$_i$ in the array.

The error reduction depends on the impurity criterion that is used, here we use Mean Squared Error. It is the impurity of the set of examples that gets routed to the internal node minus the sum of the impurities of the two partitions created by the split.

A] Technology 1: Thin Film CdTe

1. Open-Circuit Voltage (V_{oc})

module_temp	7.57E-04
tamb	2.31E-03
ape	6.89E-04
hamb	2.24E-04
inplane_irr_si	9.95E-01
rain	2.96E-04
wind_speed	1.70E-04
wind_dir	1.81E-04

2. Short-Circuit Current (I_{sc})

module_temp	4.42E-04
tamb	1.42E-04
ape	4.19E-04
hamb	1.11E-04
inplane_irr_si	9.99E-01
rain	1.25E-04
wind_speed	9.45E-05
wind_dir	1.10E-04

3. Power at MPP (P_{mpp})

module_temp	2.89E-04
tamb	1.10E-03
ape	2.56E-04
hamb	1.45E-04
inplane_irr_si	9.97E-01
rain	4.87E-04
wind_speed	1.29E-04
wind_dir	1.83E-04

B] Technology 2: Multicrystalline

1. Open-Circuit Voltage (V_{oc})

module_temp	3.10E-03
tamb	2.30E-03
ape	7.36E-03
hamb	1.22E-03
inplane_irr_si	9.84E-01
rain	7.59E-04
wind_speed	6.57E-04
wind_dir	6.53E-04

2. Short-Circuit Current (I_{sc})

module_temp	3.97E-04
tamb	1.35E-04
ape	2.53E-04
hamb	2.49E-04
inplane_irr_si	9.99E-01
rain	1.49E-04
wind_speed	1.24E-04
wind_dir	1.36E-04

3. Power at MPP (P_{mpp})

module_temp	2.94E-03
tamb	3.37E-04
ape	4.74E-04
hamb	3.33E-04
inplane_irr_si	9.95E-01
rain	3.06E-04
wind_speed	3.21E-04
wind_dir	3.18E-04

C] Technology 3: CiS substrate glass

1. Open-Circuit Voltage (V_{oc})

module_temp	5.52E-03
tamb	1.03E-03
ape	8.21E-04
hamb	3.63E-04
inplane_irr_si	9.91E-01
rain	5.04E-04
wind_speed	3.12E-04
wind_dir	2.81E-04

2. Short-Circuit Current (I_{sc})

module_temp	2.92E-03
tamb	1.07E-04
ape	2.16E-04
hamb	1.51E-04
inplane_irr_si	9.96E-01
rain	8.59E-05
wind_speed	1.01E-04
wind_dir	1.06E-04

3. Power at MPP (P_{mpp})

module_temp	3.26E-03
tamb	1.82E-04
ape	2.68E-04
hamb	1.66E-04
inplane_irr_si	9.95E-01
rain	4.51E-04
wind_speed	2.00E-04
wind_dir	1.52E-04

Tabulated above are the feature importances of each of the nine machine learning models trained.

Python code for the above RF models along with feature importance function can be found in Appendix A.

Chapter 4

Inference From Random Forest Regression Models

4.1 Results

	Open-Circuit Voltage (V_{oc})		Short-Circuit Current (I_{sc})		Power at MPP (P_{mpp})	
1: Thin Film CdTe	module_temp	7.57E-04	module_temp	4.42E-04	module_temp	2.89E-04
	tamb	2.31E-03	tamb	1.42E-04	tamb	1.10E-03
	ape	6.89E-04	ape	4.19E-04	ape	2.56E-04
	hamb	2.24E-04	hamb	1.11E-04	hamb	1.45E-04
	inplane_irr_si	9.95E-01	inplane_irr_si	9.99E-01	inplane_irr_si	9.97E-01
	rain	2.96E-04	rain	1.25E-04	rain	4.87E-04
	wind_speed	1.70E-04	wind_speed	9.45E-05	wind_speed	1.29E-04
	wind_dir	1.81E-04	wind_dir	1.10E-04	wind_dir	1.83E-04
2: Multicrystalline	module_temp	3.10E-03	module_temp	3.97E-04	module_temp	2.94E-03
	tamb	2.30E-03	tamb	1.35E-04	tamb	3.37E-04
	ape	7.36E-03	ape	2.53E-04	ape	4.74E-04
	hamb	1.22E-03	hamb	2.49E-04	hamb	3.33E-04
	inplane_irr_si	9.84E-01	inplane_irr_si	9.99E-01	inplane_irr_si	9.95E-01
	rain	7.59E-04	rain	1.49E-04	rain	3.06E-04
	wind_speed	6.57E-04	wind_speed	1.24E-04	wind_speed	3.21E-04
	wind_dir	6.53E-04	wind_dir	1.36E-04	wind_dir	3.18E-04
3: CiS substrate glass	module_temp	5.52E-03	module_temp	2.92E-03	module_temp	3.26E-03
	tamb	1.03E-03	tamb	1.07E-04	tamb	1.82E-04
	ape	8.21E-04	ape	2.16E-04	ape	2.68E-04
	hamb	3.63E-04	hamb	1.51E-04	hamb	1.66E-04
	inplane_irr_si	9.91E-01	inplane_irr_si	9.96E-01	inplane_irr_si	9.95E-01
	rain	5.04E-04	rain	8.59E-05	rain	4.51E-04
	wind_speed	3.12E-04	wind_speed	1.01E-04	wind_speed	2.00E-04
	wind_dir	2.81E-04	wind_dir	1.06E-04	wind_dir	1.52E-04

Tabulated above are the feature importances of each of the nine machine learning models trained. It is clear that primary dependency is on inplane irradiance, measured from the Si sensor, highlighted in blue. Temperature parameters, the next most significant are highlighted in orange.

From a cursory glance at the feature importance table, we infer:

1. V_{oc} is more dependent on temperature measurements than I_{sc}, for all three technologies, but more so for Multicrystalline, i.e. technology 2.
2. The most prominent feature is inplane irradiance, measured from the Si sensor.
3. Each of the technologies differ by a significant margin in their feature importances for prediction of V_{oc}, but not so much for I_{sc} and P_{mpp}.

We take a more closer look at the features and their importances below.

4.1.1 Performance Parameter 1: Short Circuit Current (I_{sc})

Thin Film CdTe

module_temp	4.42E-04
tamb	1.42E-04
ape	4.19E-04
hamb	1.11E-04
inplane_irr_si	9.99E-01
rain	1.25E-04
wind_speed	9.45E-05
wind_dir	1.10E-04

Multicrystalline

module_temp	3.97E-04
tamb	1.35E-04
ape	2.53E-04
hamb	2.49E-04
inplane_irr_si	9.99E-01
rain	1.49E-04
wind_speed	1.24E-04
wind_dir	1.36E-04

CiS Substrate Glass

module_temp	2.92E-03
tamb	1.07E-04
ape	2.16E-04
hamb	1.51E-04
inplane_irr_si	9.96E-01
rain	8.59E-05
wind_speed	1.01E-04
wind_dir	1.06E-04

The most important features for prediction of I_{sc} are given below:
1. Inplane irradiance, measured from the Si sensor.
(This is several orders of magnitude more important than the other features.)
2. Module temperature is next in significance, with more significance in CiS Substrate Glass technology than the other two.

4.1.2 Performance Parameter 2: Open Circuit Voltage (V_{oc})

Thin Film CdTe

module_temp	7.57E-04
tamb	2.31E-03
ape	6.89E-04
hamb	2.24E-04
inplane_irr_si	9.95E-01
rain	2.96E-04
wind_speed	1.70E-04
wind_dir	1.81E-04

Multicrystalline

module_temp	3.10E-03
tamb	2.30E-03
ape	7.36E-03
hamb	1.22E-03
inplane_irr_si	9.84E-01
rain	7.59E-04
wind_speed	6.57E-04
wind_dir	6.53E-04

CiS Substrate Glass

module_temp	5.52E-03
tamb	1.03E-03
ape	8.21E-04
hamb	3.63E-04
inplane_irr_si	9.91E-01
rain	5.04E-04
wind_speed	3.12E-04
wind_dir	2.81E-04

The most important features for prediction of V_{oc} are given below:
1. Inplane irradiance, measured from the Si sensor.
(This is again, several orders of magnitude more important than the other features.)
2. Temperature plays a more significant role in V_{oc} prediction than I_{sc} prediction, which is notably higher in Multicrystalline and CiS Substrate Glass technologies than in Thin Film CdTe.

The formula for V_{oc} depends linearly on Temperature, and logarithmically on Irradiance, as shown below, where I_{sc} can be assumed to be linearly dependent on Irradiance.

Current I = $I_{sc} - I_o(\frac{qV}{e^{mkT}} - 1)$

Voltage V = $V_{oc} = \frac{mkT}{q} ln(1 + \frac{I_{sc}}{I_o})$

Power P = $V[I_{sc} - I_o(\frac{qV}{e^{mkT}} - 1)]$

Where, $I_{sc} = I$ for V=0
I_o is the diode reverse bias saturation current
q is the electron charge
m is the diode ideality factor
k is the Boltzmann's constant
T is the cell temperature.

Temperature plays a more important role in deciding V_{oc} that I_{sc}, which is apparent from the distribution of importance a little more over the temperature parameters in V_{oc} than in I_{sc} prediction models. It must not go unnoticed that these features are not independent of one another. Greater value of irradiance must also directly increase the temperature of the module.

It is also clear that each of the three different technologies are more differently affected in V_{oc} than in I_{sc}, by virtue of their more significant feature importances here.

4.1.3 Performance Parameter 3: Power at Maximal Power Point (P_{mpp})

Thin Film CdTe

module_temp	2.89E-04
tamb	1.10E-03
ape	2.56E-04
hamb	1.45E-04
inplane_irr_si	9.97E-01
rain	4.87E-04
wind_speed	1.29E-04
wind_dir	1.83E-04

Multicrystalline

module_temp	2.94E-03
tamb	3.37E-04
ape	4.74E-04
hamb	3.33E-04
inplane_irr_si	9.95E-01
rain	3.06E-04
wind_speed	3.21E-04
wind_dir	3.18E-04

CiS Substrate Glass

module_temp	3.26E-03
tamb	1.82E-04
ape	2.68E-04
hamb	1.66E-04
inplane_irr_si	9.95E-01
rain	4.51E-04
wind_speed	2.00E-04
wind_dir	1.52E-04

The most important features for prediction of P_{mpp} are given below:
1. Inplane irradiance, measured from the Si sensor.
(This is again, several orders of magnitude more important than the other features.)
2. Module temperature seems to play an important role in Multicrystalline and CiS Substrate Glass technologies, while Thin Film CdTe has ambient temperature as an important factor in prediction of P_{mpp}.

4.2 Overall Findings

It can confidently be said that inplane irradiance is the most important factor in deciding performance of a solar module, irrespective of technology.

It must not go unnoticed that the Multicrystalline modules (technology 2) are quite differently affected by climate, and their performance has several branches of the trees in the forest completely to themselves.

Perhaps we may be able to generalize these results across climate regions with data trends collected from more sites. This will go on to ascertain performance of new modules in different climatic zones.

4.3 Discussion and Derivation of Arid Testing Conditions

At present, the PV industry heavily relies on the Standard Test Conditions (STC) which define a single set of values for irradiance level, temperature and spectrum for the testing of various technologies of PV modules. The real world conditions vary considerably from these stated ideal conditions, and hence the standard tests provide no real indication as to how the various PV technologies will respond to these variations in climate conditions. It is therefore essential to understand more deeply the effect of different climatic conditions on the performance of PV modules and define climate specific testing conditions.

We have studied the performance dependence of different technologies on climatic factors via super accurate models, and the features which they were built on. We may now go on to derive the climate conditions that allow best performance from all 3 technologies in arid regions of Australia, for testing in simulated environment, which may better indicate different technologies' performance in varied climates than STC. By searching for these conditions of convergence, indoor testing will better represent module performance in a new environment.

An accurate model for predicting performance parameters is useful for researchers in the Solar Energy domain, in order to correctly estimate a new module's performance. We have achieved this by way of machine learning model, built from recorded performance and meteorological data.

Chapter 5

Derivation of Arid Test Conditions

In order to define the Arid Test Conditions, representative values for the irradiance, module temperature, and solar spectrum need to be defined.

5.1 Basis for Derivation

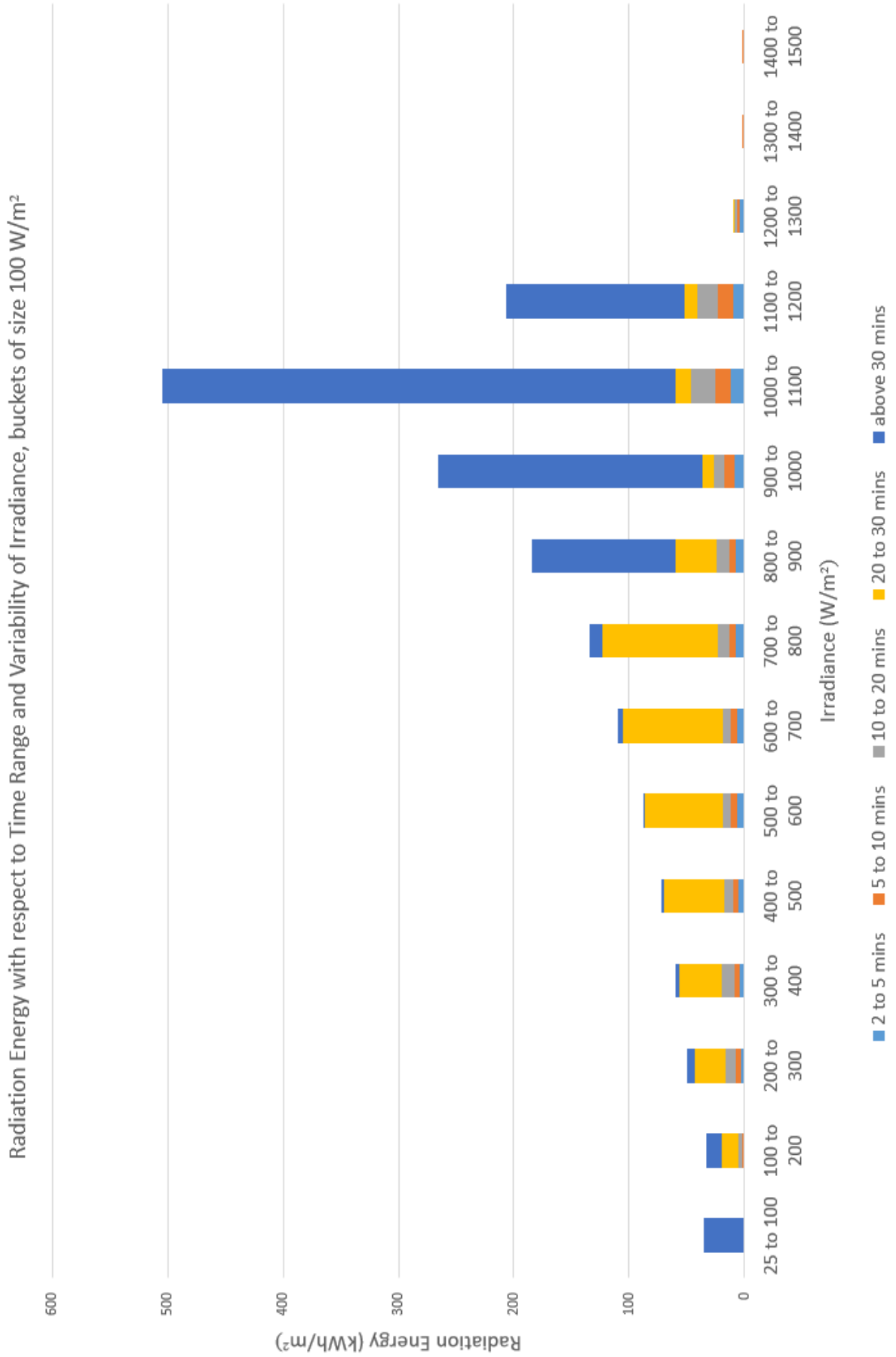
Inplane irradiance is the most important factor in deciding performance of a solar module, irrespective of technology, as deduced from the machine learning models. Bearing this in mind, we use the readings of irradiance as the primary factor in derivation of ideal test conditions.

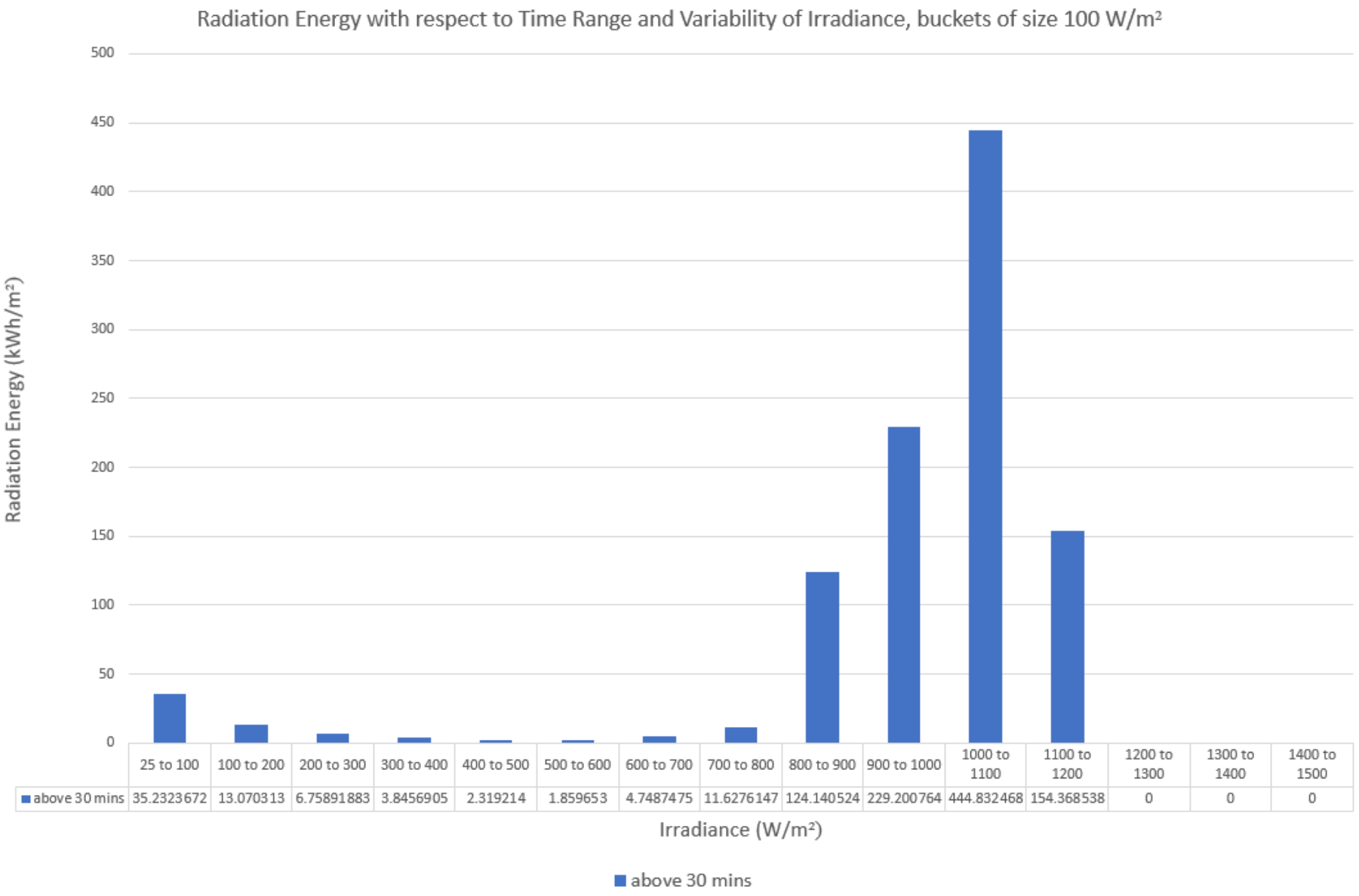
5.2 Methodology for derivation

Readings of Irradiance, Module Temperature, and calculated Average Photon Energy were taken from 1:07 PM July 24 2017 to 11:59 PM May 3 2018 at one minute intervals. The maximum reading across different sensors was taken as true, for irradiance. Irradiance below 25 W/m^2 was ignored to exclude nighttime conditions. This gave a dataset the size of 4,06,179 entries.

These readings were bucketized and considered "stable irradiance" if they lied within the same bucket for 30 minutes or longer. The bucket size was first kept at 100 W/m^2 for the purpose of determining the range of irradiance that contributed most heavily to radiation energy captured, and then zoomed in to 50 W/m^2 to decide the test condition of irradiance for Arid regions of Australia.

The graph below depicts Radiation Energy for each of the irradiance buckets of size 100 W/m^2.


Radiation Energy with respect to Time Range and Variability of Irradiance, buckets of size 100 W/m²
Radiation Energy (kWh/m²)
0
100
200
300
400
500
600
25 to 100
100 to 200
200 to 300
300 to 400
400 to 500
500 to 600
600 to 700
700 to 800
800 to 900
900 to 1000
1000 to 1100
1100 to 1200
1200 to 1300
1300 to 1400
1400 to 1500
Irradiance (W/m²)
2 to 5 mins
5 to 10 mins
10 to 20 mins
20 to 30 mins
above 30 mins


Radiation Energy with respect to Time Range and Variability of Irradiance, buckets of size 100 W/m²
Radiation Energy (kWh/m²)
500
450
400
350
300
250
200
150
100
50
0
25 to 100
100 to 200
200 to 300
300 to 400
400 to 500
500 to 600
600 to 700
700 to 800
800 to 900
900 to 1000
1000 to 1100
1100 to 1200
1200 to 1300
1300 to 1400
1400 to 1500
above 30 mins
35.2323672
13.070313
6.75891883
3.8456905
2.319214
1.859653
4.7487475
11.6276147
124.140524
229.200764
444.832468
154.368538
0
0
0
Irradiance (W/m²)
above 30 mins

It is easy to see that most of the radiation energy is caught by stable irradiance levels between 800 and 1200 W/m^2. To define the spectrum and the module temperature for Arid Testing Conditions, this range of 800 to 1200 W/m^2 is taken as input parameters.

5.3 Derivation of Module Temperature Value

The module temperature for Arid Testing Conditions was determined by the irradiance-weighted average of module temperature, for all data points with irradiance values in the considered range of 800 to 1200 W/m^2.

The resulting Arid Testing Conditions for Module Temperature = 49.4570399069 $\approx$ 50 $^\circ$ Celsius

5.4 Derivation of Spectrum Value

The spectrum value for Arid Testing Conditions was determined by the irradiance-weighted average of Average Photon Energy, for all data points with irradiance values in the considered range of 800 to 1200 W/m^2.

The resulting Arid Testing Conditions for Spectrum = 1.89606241444 $\approx$ 1.896 eV

(Python code for the derivation may be found in Appendix B.)

5.5 Derivation of Irradiance Value

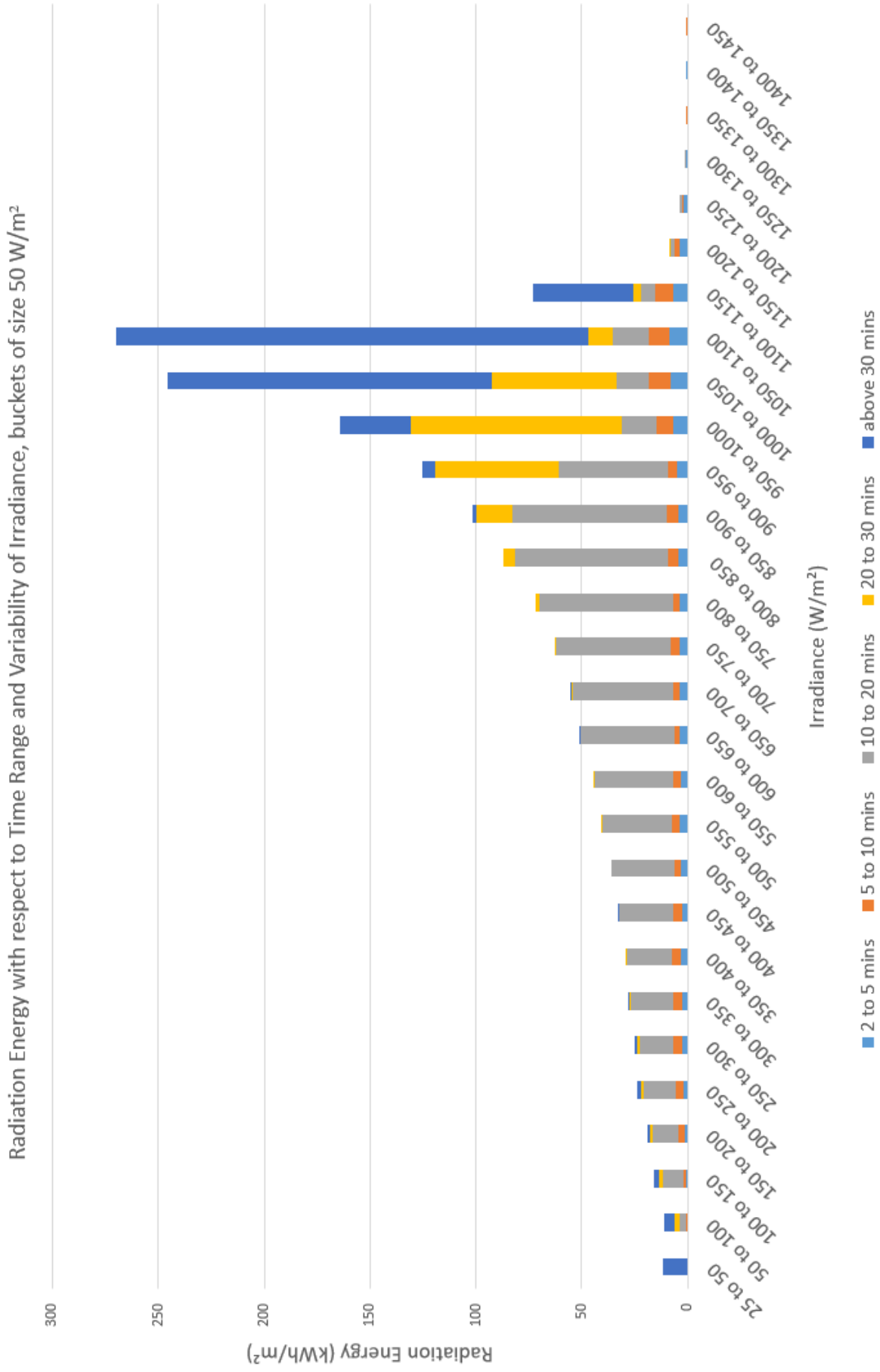
In order to derive a value for irradiance intensity which best represents the climatic zone, the bucket size was reduced to 50 W/m^2 for stable irradiance longer than 30 minutes. This revealed the range in which the most energy lay was the irradiance region between 950 and 1150 W/m^2. The mean of each of the 4 buckets in this range were weighted by the total energy contributed by them, and then averaged to give the representative value for irradiance in arid regions.

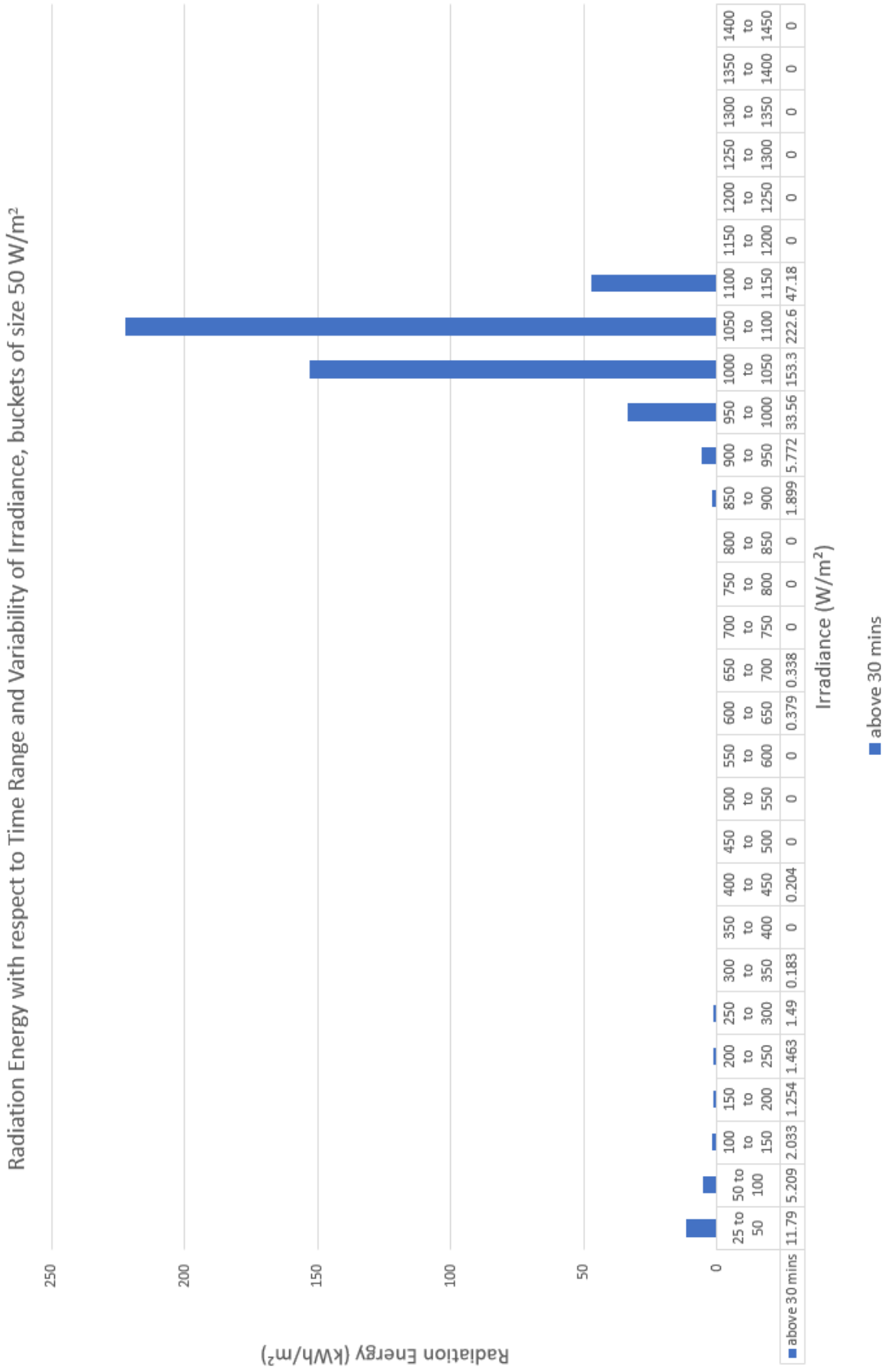
Arid Testing Conditions for Irradiance =

$$\frac{975 * 33.56 + 1025 * 153.3 + 1075 * 222.6 + 1125 * 47.18}{33.56 + 153.3 + 222.6 + 47.18}$$

= 1056.028187 $\approx$ 1055 W/m^2

The graph below depicts Radiation Energy for each of the irradiance buckets of size 50 W/m^2.


Radiation Energy with respect to Time Range and Variability of Irradiance, buckets of size 50 W/m²
Radiation Energy (kWh/m²)
300
250
200
150
100
50
0
Irradiance (W/m²)
25 to 50
50 to 100
100 to 150
150 to 200
200 to 250
250 to 300
300 to 350
350 to 400
400 to 450
450 to 500
500 to 550
550 to 600
600 to 650
650 to 700
700 to 750
750 to 800
800 to 850
850 to 900
900 to 950
950 to 1000
1000 to 1050
1050 to 1100
1100 to 1150
1150 to 1200
1200 to 1250
1250 to 1300
1300 to 1350
1350 to 1400
1400 to 1450
2 to 5 mins
5 to 10 mins
10 to 20 mins
20 to 30 mins
above 30 mins


Radiation Energy with respect to Time Range and Variability of Irradiance, buckets of size 50 W/m²
Radiation Energy (kWh/m²)
Irradiance (W/m²)
above 30 mins
0
50
100
150
200
250
25 to 50
50 to 100
100 to 150
150 to 200
200 to 250
250 to 300
300 to 350
350 to 400
400 to 450
450 to 500
500 to 550
550 to 600
600 to 650
650 to 700
700 to 750
750 to 800
800 to 850
850 to 900
900 to 950
950 to 1000
1000 to 1050
1050 to 1100
1100 to 1150
1150 to 1200
1200 to 1250
1250 to 1300
1300 to 1350
1350 to 1400
1400 to 1450
above 30 mins
11.79
5.209
2.033
1.254
1.463
1.49
0.183
0
0.204
0
0
0
0.379
0.338
0
0
0
1.899
5.772
33.56
153.3
222.6
47.18
0
0
0
0
0
0

5.6 Comparison with Tropical Test Conditions for Singapore

Previous work[1] for derivation of test conditions for tropical climate in Singapore gave the following results.

Parameter	**Standard Test Conditions (STC)**	**Tropical Test Conditions (TTC)**[1]	**Arid Test Conditions (this work)**
Irradiance (W/m^2)	1000	800	1055
Temperature ($°C$)	25	50	50
APE (eV)	1.83	1.88	1.896

It will be interesting to extend this study to the cold climate in Germany, for comparison.

Chapter 6

Conclusion and Future Work

6.1 Use of Predictive Modelling

With the advent of technology, highly accurate models with tremendous predictive capacity are in use in almost every field of science. The use of these models is not limited to prediction, but may also include remodelling and analysis of existing methodologies.

6.1.1 Random Forest and Feature Importance

In this report, the Random Forest Algorithm is first trained on a real dataset, and prediction is tuned to a very high degree of accuracy. The corresponding model is then understood, to gain insight into the features used for initial training.

6.1.2 Recommended Future Use

Of course the traditional use of the predictive model is to estimate the performance of a new module in its installed location, without actually installing it. Previous work [2] in this domain, with more complex methods has achieved comparable accuracies of prediction.

6.2 Derivation of Arid Testing Conditions

Currently, there a tremendous dependence on the Standard Test Conditions (STC) which define a single set of values for irradiance level, temperature and spectrum for the testing of various

technologies of PV modules. In reality, the meteorological and climatic conditions that these values hope to represent vary heavily from place to place. For indoor testing of new modules, the Standard Testing Conditions give no comprehensive verifiable reading of the module's performance in its installed location.

By deriving these conditions in a climate specific way, based on the most important feature obtained from the machine learning analysis, the proposed set of conditions will help estimate best case performance of a module in it's installed location.

Parameter	**Standard Test Conditions (STC)**	**Tropical Test Conditions (TTC)**[1]	**Arid Test Conditions (this work)**
Irradiance (W/m^2)	1000	800	1055
Temperature ($°C$)	25	50	50
APE (eV)	1.83	1.88	1.896

6.3 Future Work

These conditions will further be used for testing in indoor lab environments and cross verified after installing in the Arid regions of Australia.

This analysis can be repeated for the cold climates of Germany.

Appendix A

Python script for Regression using RF

```
import psycopg2 as pg
import pandas as pd
import numpy as np
from decimal import Decimal

from sklearn.ensemble import RandomForestRegressor


dbpath="host=123.456.789.123 user=postgres password=numnumnum dbname=TruePower"
try:
    conn=pg.connect(dbpath)
    print 'Database connected.'
except:
    print 'Sorry, could not connect the database. Try again with correct host, username, \
        password and DB name.'
cur=conn.cursor()


query_aus1 = """SELECT * FROM public.omt_min_meteo as omm, \
    public.module_model as mm, \
    public.module as m \
    where \
    omm.seris_id=m.seris_id and \
    m.module_model_id=mm.id and \
    mm.technology=1 and \
    omm.country='Australia' and \
    omm.module_temp is not null and \
    omm.tamb is not null and \
    omm.ape is not null and \
    omm.hamb is not null and \
    omm.inplane_irr is not null and \
    omm.inplane_irr_si is not null and \
    omm.global_irr is not null and \
    omm.global_irr_si is not null and \
    omm.direct_irr is not null and \
    omm.uv is not null and \
    omm.rain is not null and \
    omm.wind_speed is not null and \
    omm.wind_dir is not null;"""
```

```
cur.execute(query_aus1)
tech1=pd.DataFrame(cur.fetchall())
tech1.columns=['dt','hr','mn','voc','isc','pmpp','module_temp','seris_id','country','tamb', \
    'ape','hamb','inplane_irr','inplane_irr_si','global_irr','global_irr_si','direct_irr', \
    'uv','rain','wind_speed','wind_dir','id','name','technology','customer','mm.isc','mm.voc', \
    'mm.impp','mm.vmpp','pmax','dimensions','weight','datasheet','m.seris_id','module_model_id', \
    'manu_serial','remark','omt_id','string_id','site_id']
tech1 = tech1.sample(frac=1).reset_index(drop=True)

query_aus2 = """SELECT * FROM public.omt_min_meteo as omm, \
    public.module_model as mm, \
    public.module as m \
    where \
    omm.seris_id=m.seris_id and \
    m.module_model_id=mm.id and \
    mm.technology=2 and \
    omm.country='Australia' and \
    omm.module_temp is not null and \
    omm.tamb is not null and \
    omm.ape is not null and \
    omm.hamb is not null and \
    omm.inplane_irr is not null and \
    omm.inplane_irr_si is not null and \
    omm.global_irr is not null and \
    omm.global_irr_si is not null and \
    omm.direct_irr is not null and \
    omm.uv is not null and \
    omm.rain is not null and \
    omm.wind_speed is not null and \
    omm.wind_dir is not null;"""
cur.execute(query_aus2)
tech2=pd.DataFrame(cur.fetchall())
tech2.columns=['dt','hr','mn','voc','isc','pmpp','module_temp','seris_id','country','tamb', \
    'ape','hamb','inplane_irr','inplane_irr_si','global_irr','global_irr_si','direct_irr', \
    'uv','rain','wind_speed','wind_dir','id','name','technology','customer','mm.isc','mm.voc', \
    'mm.impp','mm.vmpp','pmax','dimensions','weight','datasheet','m.seris_id','module_model_id', \
    'manu_serial','remark','omt_id','string_id','site_id']
tech2 = tech2.sample(frac=1).reset_index(drop=True)

query_aus3 = """SELECT * FROM public.omt_min_meteo as omm, \
    public.module_model as mm, \
    public.module as m \
    where \
    omm.seris_id=m.seris_id and \
    m.module_model_id=mm.id and \
    mm.technology=2 and \
    omm.country='Australia' and \
    omm.module_temp is not null and \
    omm.tamb is not null and \
    omm.ape is not null and \
    omm.hamb is not null and \
    omm.inplane_irr is not null and \
    omm.inplane_irr_si is not null and \
    omm.global_irr is not null and \
    omm.global_irr_si is not null and \
    omm.direct_irr is not null and \
    omm.uv is not null and \
```

```
    omm.rain is not null and \
    omm.wind_speed is not null and \
    omm.wind_dir is not null;"""
cur.execute(query_aus3)
tech3=pd.DataFrame(cur.fetchall())
tech3.columns=['dt','hr','mn','voc','isc','pmpp','module_temp','seris_id','country','tamb', \
    'ape','hamb','inplane_irr','inplane_irr_si','global_irr','global_irr_si','direct_irr', \
    'uv','rain','wind_speed','wind_dir','id','name','technology','customer','mm.isc','mm.voc', \
    'mm.impp','mm.vmpp','pmax','dimensions','weight','datasheet','m.seris_id','module_model_id', \
    'manu_serial','remark','omt_id','string_id','site_id']
tech3 = tech3.sample(frac=1).reset_index(drop=True)

print tech1.size
print tech2.size
print tech3.size


x1 = tech1[['module_temp','tamb','ape','hamb','inplane_irr_si','rain','wind_speed','wind_dir' \
    ]].copy()
X1 = np.array(x1)
X1 -= np.mean(X1)
X1 /= np.std(X1)

y1_isc = tech1['isc'].copy()
y1_i = np.array(y1_isc)
reg1i = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg1i.fit(X1[:int(len(tech1)*0.7)],y1_i[:int(len(tech1)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg1i.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg1i.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg1i.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg1i.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg1i.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg1i.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg1i.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg1i.feature_importances_[7])
print reg1i.score(X1[int(len(tech1)*0.7):],y1_i[int(len(tech1)*0.7):])

y1_voc = tech1['voc'].copy()
y1_v = np.array(y1_voc)
reg1v = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg1v.fit(X1[:int(len(tech1)*0.7)],y1_v[:int(len(tech1)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg1v.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg1v.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg1v.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg1v.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg1v.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg1v.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg1v.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg1v.feature_importances_[7])
print reg1v.score(X1[int(len(tech1)*0.7):],y1_v[int(len(tech1)*0.7):])

y1_pmpp = tech1['pmpp'].copy()
y1_p = np.array(y1_pmpp)
reg1p = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg1p.fit(X1[:int(len(tech1)*0.7)],y1_p[:int(len(tech1)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg1p.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg1p.feature_importances_[1]),  \
```

```
    "\nape\t\t",'%.2E' % Decimal(reg1p.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg1p.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg1p.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg1p.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg1p.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg1p.feature_importances_[7])
print reg1p.score(X1[int(len(tech1)*0.7):],y1_p[int(len(tech1)*0.7):])


x2 = tech2[['module_temp','tamb','ape','hamb','inplane_irr_si','rain','wind_speed','wind_dir']].copy()
X2 = np.array(x2)
X2 -= np.mean(X2)
X2 /= np.std(X2)
y2_isc = tech2['isc'].copy()
y2_i = np.array(y2_isc)
reg2i = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg2i.fit(X2[:int(len(tech2)*0.7)],y2_i[:int(len(tech2)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg2i.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg2i.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg2i.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg2i.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg2i.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg2i.feature_importances_[5]),  \
    "\nwind_speed\t",'%.2E' % Decimal(reg2i.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg2i.feature_importances_[7])
print reg2i.score(X2[int(len(tech2)*0.7):],y2_i[int(len(tech2)*0.7):])

y2_voc = tech2['voc'].copy()
y2_v = np.array(y2_voc)
reg2v = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg2v.fit(X2[:int(len(tech2)*0.7)],y2_v[:int(len(tech2)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg2v.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg2v.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg2v.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg2v.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg2v.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg2v.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg2v.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg2v.feature_importances_[7])
print reg2v.score(X2[int(len(tech2)*0.7):],y2_v[int(len(tech2)*0.7):])

y2_pmpp = tech2['pmpp'].copy()
y2_p = np.array(y2_pmpp)
reg2p = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg2p.fit(X2[:int(len(tech2)*0.7)],y2_p[:int(len(tech2)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg2p.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg2p.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg2p.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg2p.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg2p.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg2p.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg2p.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg2p.feature_importances_[7])
print reg2p.score(X2[int(len(tech2)*0.7):],y2_p[int(len(tech2)*0.7):])


x3 = tech3[['module_temp','tamb','ape','hamb','inplane_irr_si','rain','wind_speed','wind_dir']].copy()
```

```
X3 = np.array(x3)
X3 -= np.mean(X3)
X3 /= np.std(X3)

y3_isc = tech3['isc'].copy()
y3_i = np.array(y3_isc)
reg3i = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg3i.fit(X3[:int(len(tech3)*0.7)],y3_i[:int(len(tech3)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg3i.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg3i.feature_importances_[1]),  \
    "\nape\t\t",'%.2E' % Decimal(reg3i.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg3i.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg3i.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg3i.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg3i.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg3i.feature_importances_[7])
print reg3i.score(X3[int(len(tech3)*0.7):],y3_i[int(len(tech3)*0.7):])

y3_voc = tech3['voc'].copy()
y3_v = np.array(y3_voc)
reg3v = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg3v.fit(X3[:int(len(tech3)*0.7)],y3_v[:int(len(tech3)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg3v.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg3v.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg3v.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg3v.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg3v.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg3v.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg3v.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg3v.feature_importances_[7])
print reg3v.score(X3[int(len(tech3)*0.7):],y3_v[int(len(tech3)*0.7):])

y3_pmpp = tech3['pmpp'].copy()
y3_p = np.array(y3_pmpp)
reg3p = RandomForestRegressor(max_depth=20, n_estimators = 200, random_state=0)
reg3p.fit(X3[:int(len(tech3)*0.7)],y3_p[:int(len(tech3)*0.7)])
print "module_temp\t",'%.2E' % Decimal(reg3p.feature_importances_[0]), \
    "\ntamb\t\t",'%.2E' % Decimal(reg3p.feature_importances_[1]), \
    "\nape\t\t",'%.2E' % Decimal(reg3p.feature_importances_[2]), \
    "\nhamb\t\t",'%.2E' % Decimal(reg3p.feature_importances_[3]), \
    "\ninplane_irr_si\t",'%.2E' % Decimal(reg3p.feature_importances_[4]), \
    "\nrain\t\t",'%.2E' % Decimal(reg3p.feature_importances_[5]), \
    "\nwind_speed\t",'%.2E' % Decimal(reg3p.feature_importances_[6]), \
    "\nwind_dir\t",'%.2E' % Decimal(reg3p.feature_importances_[7])
print reg3p.score(X3[int(len(tech3)*0.7):],y3_p[int(len(tech3)*0.7):])
```

Appendix B

Python script for Derivation of Arid Testing Conditions

```
import psycopg2 as pg
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dbpath="host=137.132.165.71 user=postgres password=abcd@1234 dbname=TruePower"

try:
    conn=pg.connect(dbpath)
    print 'Database connected.'
except:
    print 'Sorry, could not connect the database. Try again with correct host, username, \
    password and DB name.'

cur=conn.cursor()

query = """SELECT ot.timestamp, ot.irr, ot.modtemp, md.ape FROM public.outdoor_testing as \
    ot,public.module as m, public.meteo_data as md where m.omt_id=ot.serial_number and \
    m.site_id=1 and md.time=ot.timestamp and md.site_id=1 and ot.irr is not null order by \
    ot.timestamp asc;"""
cur.execute(query)
irr = pd.DataFrame(cur.fetchall())
irr.columns = ['timestamp', 'irr', 'modtemp','ape']

print len(irr)

i = 0
read_irr = []
read_modtemp = []
read_ape = []
while i < len(irr)-1:
    lol = irr["irr"][i]
    m = irr["modtemp"][i]
    a = irr["ape"][i]
    while i < len(irr)-1 and irr["timestamp"][i] == irr["timestamp"][i+1]:
        if lol < irr["irr"][i+1]:
```

```
            lol = irr["irr"][i+1]
            m = irr["modtemp"][i+1]
            a = irr["ape"][i+1]
            i += 1
        else:
            i += 1
    read_irr.append(lol)
    read_modtemp.append(m)
    read_ape.append(a)
    i += 1

print len(read_irr)
print len(read_modtemp)
print len(read_ape)

if irr["timestamp"][len(irr)-1] != irr["timestamp"][len(irr)-2]:
    read_irr.append(irr["irr"][len(irr)-1])
    read_modtemp.append(irr["modtemp"][len(irr)-1])
    read_ape.append(irr["ape"][len(irr)-1])
print len(read_irr)/(60*24*30)

read = np.array(read_irr)

bar_gr100 = np.around(read.astype(np.double)/100)
bar_gr50 = np.around(read.astype(np.double)/50)

print max(bar_gr100)
print max(bar_gr50)

print len(bar_gr50)
print len(read)

print irr["timestamp"][:1]

print irr["timestamp"][-1:]

less2 = [0 for x in range(15)]
two5 = [0 for x in range(15)]
five10 = [0 for x in range(15)]
ten20 = [0 for x in range(15)]
twen30 = [0 for x in range(15)]
more30 = [0 for x in range(15)]

i = 0
while i < len(read_irr)-1:
    if read_irr[i] < 25: #exclude nighttime data, and sparse radiation above
        i += 1
        continue
    else:
        mins = 1
        energy = 0
        while i<len(read_irr)-1 and bar_gr100[i] == bar_gr100[i+1]:
            mins +=1
            i += 1
            energy += read_irr[i]
        if mins<2:
            less2[int(bar_gr100[i])] += energy/60000
```

```
        elif mins<5:
            two5[int(bar_gr100[i])] += energy/60000
        elif mins<10:
            five10[int(bar_gr100[i])] += energy/60000
        elif mins<20:
            ten20[int(bar_gr100[i])] += energy/60000
        elif mins<30:
            twen30[int(bar_gr100[i])] += energy/60000
        else:
            more30[int(bar_gr100[i])] += energy/60000
        i += 1

plot100 = pd.DataFrame(
    {'less than 2 mins': less2,
     '2 to 5 mins': two5,
     '5 to 10 mins': five10,
     '10 to 20 mins': ten20,
     '20 to 30 mins': twen30,
     'above 30 mins': more30
    })

plot100.to_csv('barPlot100.csv')

less2_ = [0 for x in range(30)]
two5_ = [0 for x in range(30)]
five10_ = [0 for x in range(30)]
ten20_ = [0 for x in range(30)]
twen30_ = [0 for x in range(30)]
more30_ = [0 for x in range(30)]

i = 0
while i < len(read_irr)-1:
    if read_irr[i] < 25: #exclude nighttime data
        i += 1
        continue
    else:
        mins = 1
        energy = 0
        while i<len(read_irr)-1 and bar_gr50[i] == bar_gr50[i+1]:
            mins +=1
            i += 1
            energy += read_irr[i]
        if mins<2:
            less2_[int(bar_gr50[i])] += energy/60000
        elif mins<5:
            two5_[int(bar_gr50[i])] += energy/60000
        elif mins<10:
            five10_[int(bar_gr50[i])] += energy/60000
        elif mins<20:
            ten20_[int(bar_gr50[i])] += energy/60000
        elif mins<30:
            twen30_[int(bar_gr50[i])] += energy/60000
        else:
            more30_[int(bar_gr50[i])] += energy/60000
        i += 1

plot50 = pd.DataFrame(
```

```
    {'less than 2 mins': less2_,
     '2 to 5 mins': two5_,
     '5 to 10 mins': five10_,
     '10 to 20 mins': ten20_,
     '20 to 30 mins': twen30_,
     'above 30 mins': more30_
    })

plot50.to_csv('barPlot50.csv')

df = pd.DataFrame(
    {'irradiance': read_irr,
     'module temperature': read_modtemp,
     'ape': read_ape
    })

lt1200 = df['irradiance'] <= 1200
gt800 = df['irradiance'] >= 800
avgs = df[lt1200 & gt800]

module_temp_sum = avgs['irradiance']*avgs['module temperature']
ape_sum = avgs['irradiance']*avgs['ape']

print "Arid Testing Condition for APE =", \
    np.mean(ape_sum.astype(np.double))/np.mean(avgs['irradiance'].astype(np.double))
print "Arid Testing Condition for Module Temperature =", \
    np.mean(module_temp_sum.astype(np.double))/np.mean(avgs['irradiance'].astype(np.double))
```

Appendix C

Bibliography

1. Performance of Various Photovoltaic Module Technologies in Tropical Climate Conditions, YE Jiaying, BSc.(Microelectronics), Sun Yat-Sen University

2. Global Prediction of Photovoltaic Field Performance Differences Using Open-Source Satellite Data, Ian Marius Peters, Haohui Liu, Thomas Reindl, Tonio Buonassisi, Joule