# Using SVM in Classification

Ștefan PREDA
Economic Informatics Doctoral School
The Bucharest University of Economic Studies, Romania
spreda2003@yahoo.com

*Machine learning technology is very often used in present days. Exist many machine learning models, the theory has evolved in last year's due exponential use of those technologies. A challenge is to apply those advanced technologies in practice to solve different problems or to improve systems.*

## 1 Short theory about machine learning

Machine learning (ML) is a software that use previous data, or information, or experience it learn from the past and predict or make decisions about future data, making generalization of the existing data. In simple mode, ML predict something named output based on something named input. Input information is the known experience.

If output and input are labeled pairs, we have supervised learning [1]. Example of supervised learning is determining generated power of photovoltaic system based on weather parameters like solar irradiation level, temperature, wind speed, etc.

In case when input data is not labeled learning, process is an unsupervised learning. We can have an example in this case also from the same area, let's presume we need to classify the type of defects for a photovoltaic system based on some description of the status of components or elements that show component status, it is classification problem for determining specific groups. In [1] is mentioned also semi-supervised learning which use both approach of supervising and

unsupervised learning, we have input data labeled and unlabeled. An example in this case can be the same as previous example, but which use also for output defect classification also apart from status parameters also labeled parameters that have specific numeric values.

Input variables are named usual explanatory variables or regressors, manipulated variables [1], [2]. Output variables are named response variables or explained variable, dependent variables, measured variables. In our case response variable is generated power, explanatory variables can be: wind speed, temperature, ceiling, real feel temperature shade, real feel temperature, relative humidity, cloud cover, dew point, pressure, apparent temperature, wet bulb temperature, visibility, wind chill temperature, etc.

Sample data collection which is used for learning is named training set, collection which is used to test if estimated data is correct is named test set [1], [3]. Machine learning can do many tasks, for example:

- Classification, example ML determine if the share price will rise;
- Regression, example ML will predict generated power of a wind turbine based on wind speed.
- Clustering, putting observations in groups, for example putting mobile users in specific categories based on consuming pattern.
- Dimensionality reduction is the task when input dataset contain many parameters, this correlated with huge data set make data analyze difficult, to easy this, using dimensionality reduction, from entire set off parameters will be chosen most significant one.

## 2 About Support Vector Machine

### 2.1. SVM Classification

Support Vector Machine (SVM) is a ML that can be used for classification or regression. In [4] classification case methods start from input variables that can be assimilated like vectors {x1, x2, ..., xN}. Any of those vectors have features, for example for vector xj those are noted generic {aj1, aj2, …, ajm}. Like in real world we categorize things based on features here we try to find if based on the m features mentioned those vectors are in one class or the other(s). In simple way for two classes noted C1 and C2, the problem is to find if vectors xi pertain to C1 or C2. The easy way to separate data is using lines or hyperplanes. If our input variables xj can be separated by lines or hyperplanes, then those are linearly separable. After reading all SVM theory some time ago, I think linear way was first considered because it is easier to be mathematically explained.

If we have two input vectors, then the suggestive way to represent input data is graphically like below

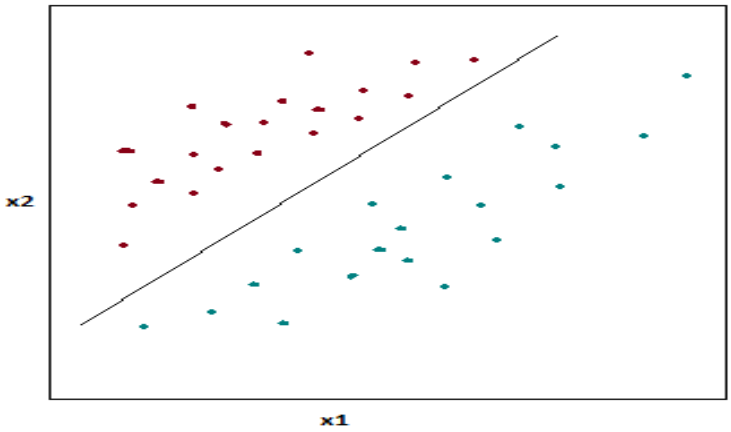


**Fig. 1.** Data separated by line in two-dimensional case

If we have 3 input vectors figure 1 will be three dimensional, and similar for n input vectors we will have "n" dimensional representation. In [5] the use of lines and hyperplanes in SVM classification is underlined more by using term linear classifiers. Basically, delimiting groups using lines or hyperplanes is straightforward and it can be computed mathematically.

We can observe that if data is separable or classified like in figure 1 between two groups there is not only a line, it is an area, which is named margin, like in Figure 2.

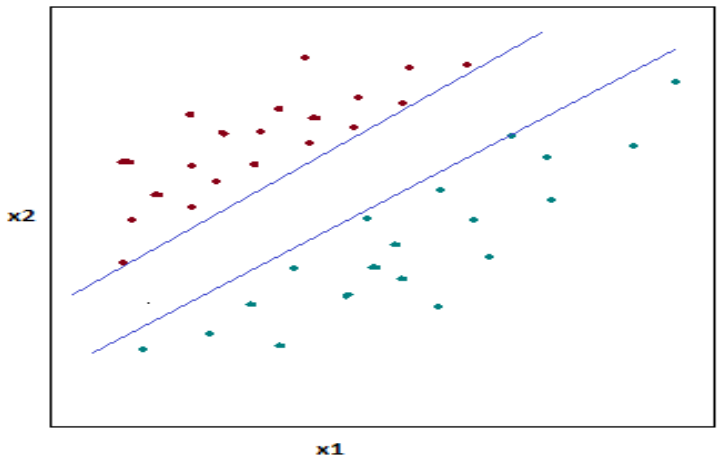


**Fig. 2.** Margin

Classification is better if the margin is wide.

The vectors that lies on edge lines or hyperplanes are named support vectors, from here the method name.

Generalizing for n vectors (n experiences, n input data) that is used in training stage of ML, the goal of SVM is to find a hyperplane that classify vectors in two classes, as there are many such solutions (or hyperplanes) the best one is that which assure the maximum size for margin.

To optimize this SVM use a function

$$S(x) = wT^* x + b;$$

where x – is an input vector

wT – is waiting vector, it is orthogonal to hyperplane and control hyperplane direction

b – is a scalar, a bias, it controls hyperplane position

For a specific input vector, SVM get that:

- It pertains to class c1 if $S(x) = wT^* x + b > 0$;
- It pertains to class c2 if $S(x) = wT^* x + b < 0$.

In real world the delimitation between objects is not always a line or a hyperplane, it can be other figure like nonlinear curves [5], [6], an for example for two-dimensional case:

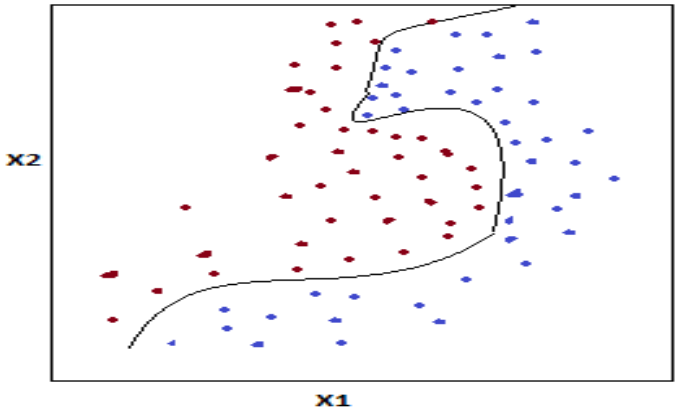


**Fig. 3.** Non-linear classification

For classification in this case SVM need a nonlinear approach, for this SVM use a kernel function, this map data into other space where classification can be done using hyperplanes [6]

There are many kernel functions, most used linear, polynomial, radial basis and sigmoid [6], [7], [8].

### 2.2 SVM regression

SVM can be used not only for classification, it can be used also for regression. A good and simple explanation for SVM principle used in regression is in [9]. Below figure describe it:

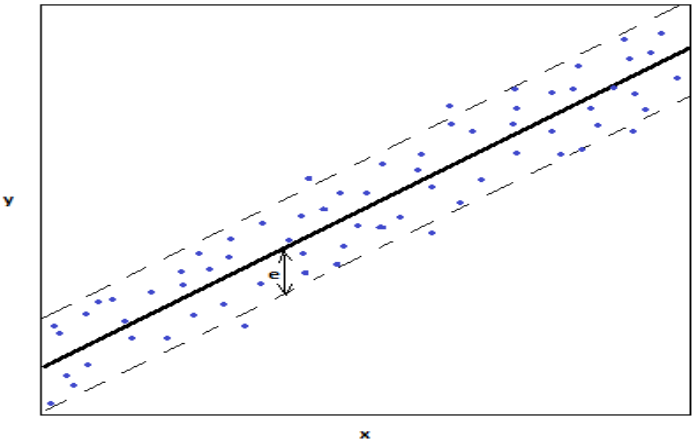


**Fig. 4.** Using SVM for regression

If in SVM classification principle is to have vectors or input data delimited by line (hyperplane) and to have a margin as big as possible, in SVM regression principle is revers, means to have vectors as much as near to the line (hyperplane), or to have vectors in a margin as narrow as possible. Margin size in SVM regression algorithm is controlled by a hyperparameter “e” as specified in figure 4.
SVM regression similar like classification can be linear or nonlinear. Also, nonlinear SVM regression have similar kernel functions: linear, polynomial, radial basis and sigmoid [10]

## 3 Example Using R for SVM classification in photovoltaic systems.

### 3.1 SVM with linear kernel

We will use data from a logger that is connected to a MPPT solar controller which is used in photovoltaic system. Target is to classify “BatteryStatus" of the battery bank based on "BatteryVoltageV", "BatterySOC" parameters all collected from logger which record data in csv or xlsx format. I will give some summary explanations in special for code that is related to SVM
First for loading necessary R libraries we use

```
library("e1071")
library("xlsx")
library("dplyr")
```

R SVM related functions are in "e1071" package
to load xlsx data logger we use:

```
setwd("C:/date")
print(getwd())
datalog2.frame <- read.xlsx("dateS2.xlsx",
sheetIndex = 1)
```

For training we will use 400 records from 476 records that are in data frame. From all data frame columns, we will use the following colomns: "BatteryVoltageV", "BatterySOC", "BatteryStatus". To fulfil those later we will use variables:

```
strain <- sample(476, 400)
head(strain, 5)
coloane <- c("BatteryVoltageV",
"BatterySOC", "BatteryStatus")
```

Data frame named datalog2.frame contain all parameters recorded by logger, because we will use in analyze 3 parameters we will extract those from data frame and create new data frame named datalog3.frame.

```
datalog3.frame <-
data.frame(datalog2.frame$BatteryVoltage
V, datalog2.frame$BatterySOC,
datalog2.frame$BatteryStatus)
names(datalog3.frame)[1:3] =
c("BatteryVoltageV", "BatterySOC",
"BatteryStatus")
head(datalog3.frame, 1)
```

For training we will we will create a new data frame named datalog3_train.frame it contain 400 records.

```
datalog3_train.frame <-
datalog3.frame[strain, coloane]
```

For test we will use the remaining 76 records from 476

```
stest <- datalog3.frame[-strain, coloane]
head(stest, 2)
```

To create SVM model we will use [12], [13], [14], [15]

```
mysvmlearn <- svm(BatteryStatus ~ ., data
= datalog3_train.frame, kernel = "linear",
cost = 0.1, scale = FALSE)
```

To plot classification, we will use [12], [13], [14], [15]:

```
plot(mysvmlearn, datalog3_train.frame[,
coloane])
```

```
R Console
> setwd("C:/date")
> print(getwd())
[1] "C:/date"
> datalog2.frame <- read.xlsx("dateS2.xlsx", sheetIndex = 1)
> strain <- sample(476, 400)
> head(strain, 5)
[1]  76 336 411 345 157
> coloane <- c("BatteryVoltageV", "BatterySOC", "BatteryStatus")
> datalog3.frame <- data.frame(datalog2.frame$BatteryVoltageV, datalog2.frame$BatterySOC, datalog2.frame$BatteryStatus)
>
> names(datalog3.frame)[1:3] = c("BatteryVoltageV", "BatterySOC", "BatteryStatus")
>
> head(datalog3.frame, 1)
  BatteryVoltageV BatterySOC BatteryStatus
1           25.96         69        Normal
>
> datalog3_train.frame <- datalog3.frame[strain, coloane]
>
> stest <- datalog3.frame[-strain, coloane]
>
> head(stest, 2)
  BatteryVoltageV BatterySOC BatteryStatus
2           25.99         70        Normal
3           26.07         71        Normal
> mysvmlearn <- svm(BatteryStatus ~ ., data = datalog3_train.frame, kernel = "linear", cost = 0.1, scale = TRUE)
>
> plot(mysvmlearn, datalog3_train.frame[, coloane])
> q()
> mysvmlearn <- svm(BatteryStatus ~ ., data = datalog3_train.frame, kernel = "linear", cost = 0.2, scale = TRUE)
> plot(mysvmlearn, datalog3_train.frame[, coloane])
> mysvmlearn <- svm(BatteryStatus ~ ., data = datalog3_train.frame, kernel = "linear", cost = 0.3, scale = TRUE)
> plot(mysvmlearn, datalog3_train.frame[, coloane])
>
```

**Fig. 5.** Run SVM for classification

Based on [11], [12] when build model and use SVM function in R an important parameter is the cost. Cost parameter control margins and training errors. If cost is small this can generate big margins and make SVM susceptible to errors, revers if cost is big errors are less and margins are small.

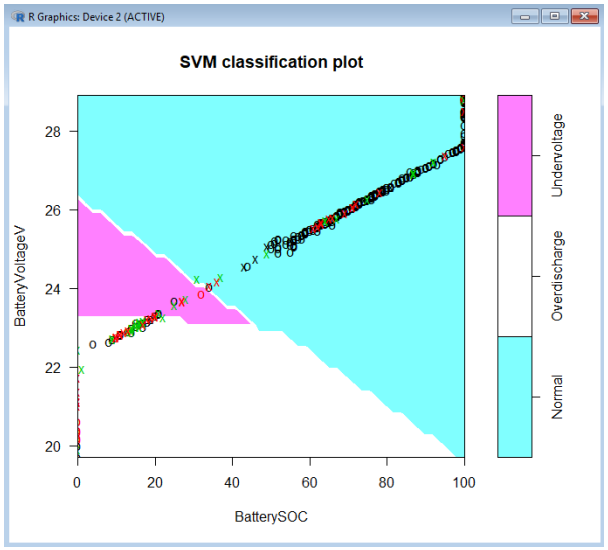


**Fig. 6.** Plot for cost=0.3

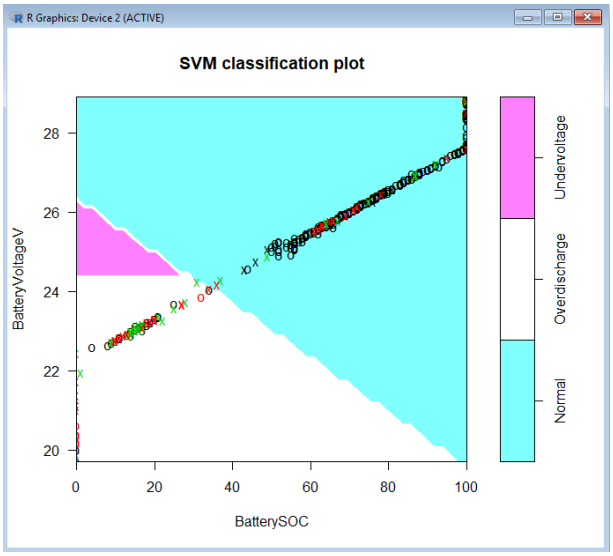


**Fig. 7.** Plot for cost=0.2

```
datatuned <- tune(svm, BatteryStatus ~ .,
data = datalog3_train.frame, kernel =
"linear", ranges = list(cost = c(0.001, 0.01,
0.1, 0.2, 0.3, 0.5, 1, 5, 10, 50, 100)))
summary(datatuned)
```

This will show:

```
> summary(datatuned)
Parameter tuning of ‘svm’:
- sampling method: 10-fold cross
validation
- best parameters:
 cost
 0.01
```

```
- best performance: 0.1128846
- Detailed performance results:
   cost     error dispersion
1  1e-03 0.2205128 0.06417731
2  1e-02 0.1128846 0.05589104
3  1e-01 0.1453205 0.04362853
4  2e-01 0.1528846 0.04157534
5  3e-01 0.1503846 0.04084294
6  5e-01 0.1478846 0.03814453
7  1e+00 0.1478846 0.03814453
8  5e+00 0.1528846 0.04321340
9  1e+01 0.1453205 0.04200667
10 5e+01 0.1428205 0.04404488
11 1e+02 0.1353205 0.04434103
```

We see the best performance is considered for cost=0.01, it has smallest error. However, we see also that for bigger cost like 50, 100 error decrease and also dispersion decrease, it means that tune function should be run repeatedly for other ranges that include this trend.

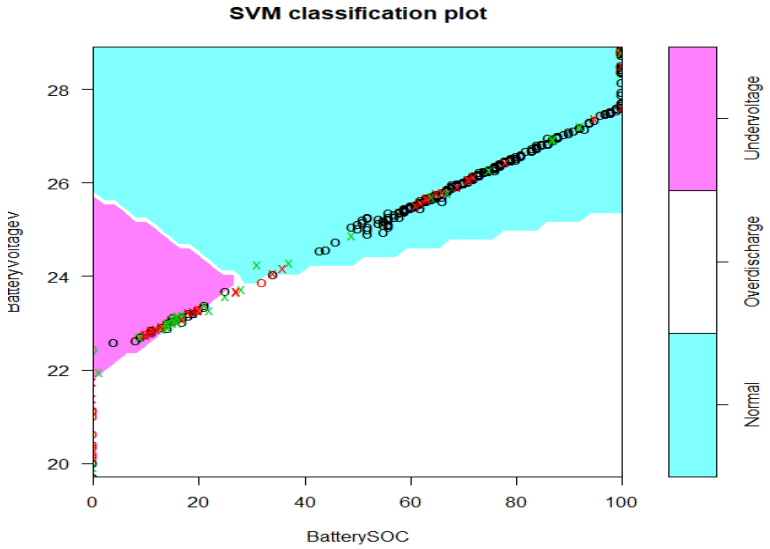


**Fig. 8.** Plot for cost=100

It is interesting to observe that for cost=100 not only that error and dispersion decrease but also classification is more appropriate by reality, means it show normal area to be area where BateryVoltageV is higher, which is good.

To predict we will use with predict function generated model SVM data frame, named mysvmlearn, and test data:

```
 predict.frame <- predict(mysvmlearn,
stest, type = "class")
```

To display standard confusion matrix, we will use

```
> table(stest[, 1], predict.frame)
  predict.frame
      Normal Overdischarge Undervoltage
  22.24      0           0           1
  22.74      0           0           1
  22.85      0           0           1
  22.87      0           0           1
  22.9       0           0           1
  22.91      0           0           1
  23         0           1           1
  23.04      0           0           1
  23.05      0           0           1
  23.12      0           0           1
  23.17      0           0           1
  23.25      0           0           1
  23.26      0           0           1
  23.73      0           1           0
……..lines omitted ……..
  28.33      1           0           0
  28.34      1           0           0
  28.51      1           0           0
  28.85      3           0           0
>
```

The interpretation of confusion matrix in this case is smoothly.
For example, first line
22.24 0 0 1
There is 1 record that is classified to be Undervoltage, this seems ok like interpretation

For line
28.34 1 0 0
It means there is 1 record that is classified to be Normal, this also seems ok.

### 3.2. SVM with polynomial kernel

We will use SVM with polynomial kernel, in the same time with model build we will run tune function to get best parameters for model. Commands are:

```
polinomtune = tune.svm(BatteryStatus ~ .,
data=datalog3_train.frame,
kernel="polynomial", degree=c(3,4,5),
coef0=c(0.1,0.5,1,2,3,4))
> summary(polinomtune)
```

Response is:

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross
validation
- best parameters:
 degree coef0
      3   0.5
- best performance: 0.09525641

- Detailed performance results:
   degree coef0      error dispersion
1       3   0.1 0.09782051 0.03455756
2       4   0.1 0.10544872 0.04612342
3       5   0.1 0.10044872 0.04647723
4       3   0.5 0.09525641 0.03296725
5       4   0.5 0.09782051 0.04183038
6       5   0.5 0.09794872 0.04862114
7       3   1.0 0.10794872 0.04495231
8       4   1.0 0.09532051 0.03901788
9       5   1.0 0.10294872 0.04562073
10      3   2.0 0.10282051 0.04009153
11      4   2.0 0.10532051 0.03892650
12      5   2.0 0.10544872 0.04612342
13      3   3.0 0.10532051 0.03709964
14      4   3.0 0.10532051 0.03892650
15      5   3.0 0.10288462 0.04208615
16      3   4.0 0.10532051 0.03709964
17      4   4.0 0.10538462 0.04097352
18      5   4.0 0.10294872 0.04246732
```

The best parameters for model are degree=3 and coef0=0.5
Running model for degree=3 and coef0=0.5 and representing graphic we have :

```
dataframePolinom=svm(BatteryStatus ~ .,
data=datalog3_train.frame,
kernel="polynomial", degree=3,
coef0=0.5)
plot(dataframePolinom,
datalog3_train.frame[, coloane])
```

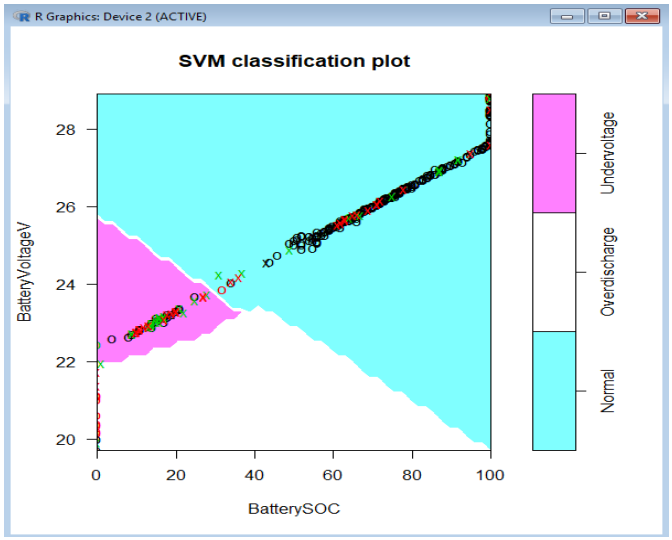


**Fig. 9.** SVM run with polynomial kernel

## 4 Conclusions

SVM was used in current example for a photovoltaic system to classify data related to battery banc, results obtained are verified in practice, we saw for example in the interpretation of confusion matrix for SVM with polynomial kernel that results match battery voltage and status from real life.

Also, SVM classification either linear or polynomial kernel, need tuning to detect best parameters that assure for model a good accuracy. In practice it can be necessary even to run tune function with many range of parameters such as to fine a combination of parameters that a re suitable for use case.

SVM classification results offer good graphical results that can be used in practice, graphs can be used to monitor systems in time. For example, from figure 7 can be created for data sets for very week and those can be compared, in this way we have information if something changes in system related to this component. Referring to this in [16] is specified that monitoring system by comparing graphs this is one of the best recommendation

## References

[1] Raúl Garreta; Guillermo Moncecchi; Trent Hauck; Gavin Hackeling, scikit-learn : Machine Learning Simplified, Packt Publishing, Pub. Date: November 10, 2017, Web ISBN-13: 978-1-78883-152-9

[2] Henrik Brink Joseph W. Richards Mark Fetherolf, Real-World Machine Learning, Manning Publications, September 20, 2016, Print ISBN-13: 978-1-61729-192-0

[3] Scott V. Burger, Introduction to Machine Learning with R, O'Reilly Media, Inc., March 21, 2018, Print ISBN: 978-1-4919-7644-9

[4] Machine Learning, Mohssen Mohammed; Muhammad Badruddin Khan; Eihab Mohammed Bashier, CRC Press, August 19, 2016, Print ISBN-13: 978-1-4987-0538-7

[5] Jason Bell, Machine Learning: Hands-On for Developers and Technical Professionals, John Wiley & Sons, November 3, 2014, Print ISBN-13: 978-1-118-88906-0

[6] Himani Bhavsar, Mahesh H. Panchal, A Review on Support Vector Machine for Data Classification, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 1, Issue 10, December 2012

[7] Arti Patle, Deepak Singh Chouhan, SVM kernel functions for classification, Advances in Technology and Engineering (ICATE), 2013 International Conference, IEEE Xplore: 06 June 2013

[8] Hussein A Kazem, Jabar H. Yousif, Miqdam T Chaichan, Modelling of Daily Solar Energy System Prediction using Support Vector Machine for Oman, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 20 (2016) pp. 10166-10172 © Research India Publications.

[9] Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media, Inc., March 30, 2017

[10] Giuseppe Ciaburro, Regression Analysis with R, Publisher: Packt Publishing, January 31, 2018, Print ISBN-13: 978-1-78862-730-6

[11] Kuntal Ganguly, R Data Analysis Cookbook - Second Edition, Packt Publishing, September 20, 2017, Web ISBN-13: 978-1-78712-531-5

[12] Viswa Viswanathan; Shanthi Viswanathan; Atmajitsinh Gohil; Chiu Yu-Wei, R: Recipes for Analysis,

Visualization and Machine Learning, Packt Publishing, November 24, 2016, Print ISBN-13: 978-1-78728-959-8

[13] Cory Lesmeister, Mastering Machine Learning with R, Packt Publishing, October 28, 2015, Print ISBN-13: 978-1-78398-452-7

[14] Raghav Bali; Dipanjan Sarkar; Brett Lantz; Cory Lesmeister, Unleash Machine Learning Techniques, Packt Publishing, October 24, 2016, Web ISBN-13: 978-1-78712-828-6

[15] Cory Lesmeister, Mastering Machine Learning with R - Second Edition, Packt Publishing, April 24, 2017, Print ISBN-13: 978-1-78728-747-1

[16] IEA International Energy Agency, Photovoltaic and Solar Forecasting: State of the Art, Report IEA PVPS T14-01:2013.

Stefan PREDA graduated the Faculty of Economic Cybernetics, Statistics and Informatics, with a bachelor degree in Economic Informatics in 2013. In 2015 he got his master degree from the same faculty of the Bucharest University of Economic Studies, specialization in Databases Support for Business. Currently he is working on Oracle Corporation, like Senior Principal Technical Support Engineer, software analyst, in Fusion Middleware, EMEA Identity Management team.