Optimization
Iranian Journal of Optimization
Volume 13, Issue 1, 2021, 69-79
Research Paper

Online version is available on: www.ijo.rasht.iau.ir

Islamic Azad University
Rasht Branch
ISSN: 2588-5723
E-ISSN:2008-5427

Application of DEA to Measure the Efficiency of Open Source Software Projects

Ehsan Zanboori[*1], Fateme Rostami[2], Saeid Ghobadi[3]
[1]Department of Mathematics, Nourabad Mamassani Branch, Islamic Azad University, Nourabad Mamassani, Iran
[2]Master Student, Nourabad Mamassani Branch, Islamic Azad University, Nourabad Mamassani, Iran
[3]Department of Mathematics, Khomeinishahr Branch, Islamic Azad University, Isfahan, Iran

Revise Date: 11 March 2021
Accept Date: 20 August 2021

Abstract

This paper evaluates the relative performance of open source software projects by evaluating multiple project inputs and multiple project outputs by using data envelopment analysis (DEA) model. The DEA model produces an efficiency score for each project based on project inputs and outputs. One of the important issues in data envelopment analysis is ranking DMUs. In this paper, open source software projects (OSS) are considered as decision making units which consume inputs to generate outputs. In this article, three standard Data Envelopment Analysis (DEA) models are used to evaluate the open source software projects. Also, super-efficiency model are used for ranking. Due to the inability of the models to rank projects, the AP-super efficiency model (the most important and popular method for ranking units) has been used for ranking OSS projects. The result of this research is a practical model that can be used by OSS project developers in order to evaluate the relative performance of their projects and make decision for their sources. Also, OSS projects can now be adequately ranked and evaluated according to project performance.

Keywords:
Computer software
OSS projects
DEA
Ranking

***Correspondence E-mail**: e.zanboori@gmail.com

INTRODUCTION

Open source software (OSS) production activities deviate significantly from conventional software engineering practices. In 1998, a group of programmers and software exports announced that the term "Open source software" should be replaced with a less ambiguous one, which is more convenient for the organizational world, "Free software", Crowston and Shamshurin (2017). Open source software development practices represent a significant difference of traditional software engineering approaches. Large groups of globally distributed developers, parallel development facilitated by highly modular code, independent peer review, timely input to developers, high levels of user participation, and rapid release schedules define the OSS development process in general, Kalina and Czyzycki (2005). Although open-source software developers often use software development related tools, there is some evidence that indicates open source software development societies do not accept many (several) modern software engineering processes, Pereira (2021). Over the years, the productivity of software development projects has become a topic of interest. In spite of having success in many open source software projects, many of these projects became inactive and even did not keep working. Payne investigated the security of open source software projects, Payne (2002). Jorgensen (2001) identified an open source software development lifecycle located at the boundaries of the implementation phase of traditional software development, including code development, code review, pre-test, development release, development debugging and product presentation.

Paradi et al. (1997) used the previous data envelopment analysis studies to incorporate two new project output scales, which were non-relative quality scales, and software time scales (Paradi et al., 1997; Zanboori et al., 2014). This study highlights the problems of software due to ignoring quality while evaluating software project performance. Finally, the authors explain that data envelopment analysis can be an effective management tool for improving software production. Applying data envelopment analysis to software project performance, Filitman (2003) also considered additional project output measures.

In particular, the authors explicitly examined the complexity of output by determining the number of locations, the number of business units, and the number of simultaneous users in each project. These results were used to create additional methods to determine whether a new proposed project has compatible features with a set of efficient projects. They used a multidimensional output size that includes the number of users, sites, factories, companies, relationships, EDIs (Electronic Data Interchange), conversions, changes, modules, and reports. In particular, they conclude that there were significant differences between software projects in different industries, Stensrud and Myrtveit (2003).

Various theoretical and practical methods and models have been proposed for ranking units as one of the main issues in the DEA literature, Heidary et al. (2018), Zanboori et al (2014). One of the recent references in DEA literature, Hosseinzadeh et al. (2013), categorized the following seven different groups for ranking DMUs: (i) cross-efficiency; (ii) finding optimal weights; (iii) super-efficiency; (iv) benchmarking; (v) multivariate statistics in DEA; (vi) multi-criteria decision making (MCDM) methodologies and DEA; and (vii) other techniques including Monte Carlo method. In this article, the OSS projects first are evaluated by applying standard DEA models. Then, super efficiency model is applied to rank the projects. This ranking is helpful to compare the software. The results show that DEA can be used as an effective tool for analyzing the performance of soft wares. The super efficiency method has some advantages and disadvantages compared to other ranking methods, and of course, it has been used in this article due to its simplicity of use. Other researchers can use other methods to rank OSS projects.

The efficiency of software development initiatives has attracted people's interest over the years. The rise of the OSS development process necessitates a reconsideration of what performance means in the context of a software development project. The goal of this study is to create and evaluate a model of the relative performance of OSS projects. This OSS development efficiency model will take into account both the unique aspects of OSS software development as well as the existing body of research on software productivity and efficiency applied to traditional software engineering methodologies.

This paper is organized as follows. Section 2 and 3 contain a brief literature review of open source software and data envelopment analysis. Evaluation and ranking of open source software are discussed in Section4. Conclusion will end the paper.

OPEN SOURCE SOFTWARE

Open source software projects allow users to use software for any purpose freely. These codes can be studied, modified and redistributed freely, Fitzgerald (2018). Although open source software is free, the profit potential of open source software projects for software development companies is enormous. Investors have invested approximately $ 400 million in 50 open source companies in the 18 months leading up to 2005 Lacy (2005). These products are satisfying for customers' needs, business and birth of profitable companies such as Pentaho, SugarCRM, GreenPlum. These companies are building a new generation of commercial applications for the web content management, customer relationships and organizational resources that are cheaper and many of them are more dynamic than their business counterparts. The potential financial benefit for an open source software developer comes from the supporting and maintaining the add-on features they can provide for their products. The success of open source software projects is attributed to the quality, portability and scalability of the software product and the commitment, expertise, and speed of software developers' progress Gao et al. (2021).

Software developers may want to publish their software under an open license so that anyone else can create or being able to be aware of the content of the same software. Using open source software in general, anyone would be able to create certain terms in software, transfer it to new operating systems, share it with others, or in some cases, retrieve it. The basic reasons for using open source software are as following:

1. Security
2. Being reasonable
3. Transparency
4. Durability
5. Interoperability with other software

Open source software development methods represent a significant difference from traditional software engineering approaches. . In general, the open source software development process has been defined by the large communities (groups) of developers in all over the world. Parallel (synchronous) development is specified by modulation codes (Zanboori et al.,2014). While open source software developers often use software development related tools. There is evidence that open source software development societies do not accept many (many) modern software engineering processes (Zanboori et al.,2014). Over the years, productivity of software development projects has become a topic of interest. One of the best examples of open source software is the Linux operating system, which is currently the safest operating system set up on most servers in the world Monteiro et al. (2020). This software codes are accessible as well as the most popular mobile operating system, the Android operating system, the popular telegram messenger. Moreover, Word-Press which is the most powerful content management system in the world. The benefits of open source software include free software, community creativity, more plug-ins and better bugs. In addition, the most important disadvantages of this type of software can be the lack of technical support and the possibility of abuse.

DATA ENVELOPMENT ANALYSIS (DEA)

Data envelopment analysis (DEA) is a non-parametric mathematical programming model considers multiple inputs and outputs of decision-making units (DMUs). This technique is considered with comparative assessment of the efficiency of decision making units. In the classical DEA models, the efficiency of a DMU is obtained by maximizing ratio of the weighted sum of its outputs to the weighted sum of its inputs, subject to the condition that this ratio does not exceed one for any DMU. In other words, the efficiency of the DMUs could be measured by DEA technique from the optimistic viewpoint. This technique has been used and developed by many scholars in the different framework, Heidary (2018). DEA has demonstrated to be an effective technique for measuring the relative efficiency of a set of homogeneous DMUs which utilize the same inputs to produce the same outputs. To describe the DEA efficiency measurement, assume that there are n DMUs, $\{DMU_j | j = 1,2,\dots,n\}$, and each DMU_j generates s outputs $y_{rj}, (r = 1,2,\dots,s)$, by utilizing m inputs $x_{ij}, (i = 1,2,\dots,m)$. Equipped with the postulates Feasibility, Unbounded Ray, Convexity, Free disposability and minimal extrapolation, the production possibility set (PPS) spanned by all DMUs is defined as follows:

$$P = \left\{ \begin{array}{l} (x_1,\dots,x_m,y_1,\dots,y_s) \Big| x_i \geq \sum_{j=1}^{n} \lambda_j x_{ij}, i = 1,\dots,m, \\ y_r \leq \sum_{j=1}^{n} \lambda_j y_{rj}, r = 1,\dots s, \lambda_j \geq 0 \end{array} \right\}.$$

Based on the proposed PPS, both radial and non-radial models are proposed. In radial models, which are also known as CCR and BCC models, measure the efficiency of DMU_o by maximizing the ratio of its weighted sum of outputs to its weighted sum of inputs. Radial models deal with proportional changes of inputs or outputs.

In the continuation of this section, CCR, BCC, SBM models described where CCR and BCC models are radial models, SBM-model is non-radial.

Radial models

As stated before, efficiency is described as proper working under the influence of organizational indicators such as earnings per unit, sales per unit, and so on, which can be expressed as the output to input ratio.

$$\text{Efficiency} = \frac{\text{outputs}}{\text{inputs}}$$

The CCR model is the first DEA model that was used for calculating the efficiency of DMUs. This model was presented in 1978 by Charnes et al[1]. (1978) and is as follows:

$$\begin{aligned} & Min: \ \theta \\ & s.t.: \ \sum_{j=1}^{n} \lambda_j x_{ij} \leq \theta x_{ip}, \quad i = 1,\dots,m, \qquad (1) \\ & \sum_{j=1}^{n} \lambda_j y_{rj} \geq y_{rp}, \quad r = 1,\dots,s, \\ & \lambda_j \geq 0, \qquad j = 1,\dots,n, \end{aligned}$$

This model is a constant return to scale (CRS) program and it assumes that the status of all input/output variables are known prior to solving the model. The efficiency ratio θ_o ranges between zero and one, with DMU_p being considered relatively efficient if it receives a score of one. From a managerial perspective, this model delivers assessments and targets with an output maximization orientation. If the constraint $\sum_{j=1}^{n} \lambda_j = 1$, is added up to model 1, the resulting model is known as BCC (Banker, Charnes, Cooper, 1984). The production frontier of this model spanned by the convex hull of the existing DMUs. The input-oriented BCC model evaluates the efficiency of DMU_p , $(p = 1,2,\dots,n)$ by solving the following linear program:

[1] Charnes, Cooper and Rhodes (CCR)

$$
\begin{aligned}
&Min:\ \theta \\
&s.t.:\ \sum_{j=1}^{n}\lambda_j x_{ij} \le \theta x_{ip}, \quad i=1,...,m, \qquad (2)\\
&\sum_{j=1}^{n}\lambda_j y_{rj} \ge y_{rp}, \quad r=1,...,s,\\
&\sum_{j=1}^{n}\lambda_j = 1,\\
&\lambda_j \ge 0, \qquad j=1,...,n,
\end{aligned}
$$

This model is a variable return to scale (VRS) program and it assumes that the status of all input/output variables are known prior to solving the model. The efficiency ratio θ_p ranges between zero and one, with DMU_p being considered relatively efficient if it receives a score of one. Otherwise, the unit will be considered as an inefficient unit.

Non-radial models

Slacks-based measure (SBM) models, put aside the assumption of proportional changes in inputs and outputs, and deal with slacks directly. This non-radial model has three variations, namely input, output and non-oriented. The non-oriented model is both input-oriented and output-oriented. The non-oriented or both-oriented SBM efficiency is defined by solving the following model:

$$
\begin{aligned}
&Min:\ \rho = \frac{1-\frac{1}{m}\sum_{i=1}^{m}\frac{z_i^-}{x_{ik}}}{1+\frac{1}{s}\sum_{r=1}^{s}\frac{z_r^+}{y_{rk}}}\\
&s.t.:\ \sum_{j=1}^{m}\lambda_j x_{ij} + z_i^- = x_{ik}, \quad i=1,...,m, \qquad (3)\\
&\sum_{j=1}^{m}\lambda_j y_{rj} - z_r^+ = y_{rk}, \quad r=1,...,s,\\
&\lambda_j \ge 0, \qquad j=1,\cdots,n\\
&z_i^- \ge 0, \qquad i=1,\cdots,m\\
&z_r^+ \ge 0, \qquad r=1,\cdots,s
\end{aligned}
$$

Slacks-Based Measure (SBM) model which was introduced by Tone (2001) put aside the assumption of proportionate changes in inputs and outputs, and deal with slacks directly. DMU_o is defined as an SBM-efficient unit whenever $\rho^* = 1$ (or $z_i^{-*} = z_i^{+*} = 0$).

Super efficiency model

In spite of the advantageous and wide application in DEA models, there are still some shortcomings in DEA evaluation. However, the traditional self-evaluated DEA models with total weight flexibility may evaluate many DMUs as DEA efficient and cannot make any further distinction among these efficient DMUs. Therefore, one of the main shortfalls of the traditional DEA model (CCR or BCC model) is its inability to discriminate among DMUs that are all deemed efficient. To improve the power of discriminating the efficient DMUs, Andersen and Petersen (1993) proposed the super-efficiency models which is one of the most important and popular methods for ranking units. The basic idea of the proposed method is of leaving out one unit and assessing efficiency by the remaining units. Then, the model tried to gain minimum distance between the eliminated unit of PPS and the efficiency boundary. In other words, this method is based on eliminating units under evaluation of PPS and establishing a new PPS to evaluate them. Again imagine there are n DMUs, $\{DMU_j | j = 1,2,\dots,n\}$, in which each DMU have multiple outputs $y_{rj}, (r = 1,2,\dots,s)$, and multiple inputs $x_{ij}, (i = 1,2,\dots,m)$. By eliminating the under evaluated unit from PPS, the new PPS is as follows:

$$
P_c' = \left\{ \begin{array}{l} (x_1,...,x_m,y_1,...,y_s) \Big| x \ge \sum_{j=1,j\ne k}^{n}\lambda_j x_j, \\ y \le \sum_{j=1,j\ne k}^{n}\lambda_j y_j, \lambda_j \ge 0, j \ne k \end{array} \right\}
$$

Equipped with the new PPS, Anderson and Peterson (1993) proposed super efficiency (AP model) as follows:

$$
\begin{aligned}
&Min:\ \theta\\
&s.t.:\ \sum_{j=1,j\ne p}^{n}\lambda_j x_{ij} \le \theta x_{ip}, \quad i=1,...,m,\\
&\sum_{j=1,j\ne p}^{n}\lambda_j y_{rj} \ge y_{rp}, \quad r=1,...,s, \qquad (4)\\
&\lambda_j \ge 0, \qquad j=1,...,n, j\ne p
\end{aligned}
$$

(4)

In spite of its advantages and wide applications, there are still some shortcomings in DEA super efficiency. Infeasibility and unboundedness may reduce the usefulness of this method. As mentioned in the introduction section, there are other ranking methods, but in this article, the super efficiency ranking method will use.

OPEN SOURCE SOFTWARE PERFORMANCE

Identifying high-performing projects and capitalizing on the best practices associated with these projects can help organizations improve their software development methodologies. The usage of DEA is a frequent approach of benchmarking and analyzing the relative performance of software projects. DEA operational units are commonly defined as software projects in the software management literature. The data set for this study consists of 34 open source software projects listed on sourceforge.net. Only highly ranked projects in the security domain were considered to improve the homogeneity of the projects under consideration. By concentrating on a single domain, in this case security, the possibility of comparing software projects from disparate domains is limited. The possibility of considering rogue projects with little traction in the OSS community is eliminated when only highly ranked projects are considered.
This study assesses the relative success of security-based OSS projects by determining how effectively multiple project inputs generate multiple project outputs. The number of individual users who have submitted software bugs and the total number of developers for a project are the two inputs considered for the 34 projects.

In this paper, 34 security-based open source software projects have been evaluated by using of BCC, SBM and AP-super efficiency models. Each project includes two inputs (number of developers and number of error senders) and three outputs (rank, number of downloads, and number of bytes per download).

Table 1: Data of 34 open source softwares

DMU Issue	project name	Bug {I}	Developers {I}	Rank {O}	Downloads {O}	K per download {O}
DMU 1	Open Computers and software Inventory	66	7	4	39,425	35,510
DMU 2	Endian Firewall	24	10	18	9,033	106,310
DMU 3	KeePass Password Safe	44	8	27	82,098	688
DMU 4	Password Safe	64	29	34	34,925	1,354
DMU 5	Ophcrack	1	4	36	66,146	338,645
DMU 6	IP Cop Firewall	83	39	46	119,097	35,265
DMU 7	Clam Win Free Anivirus	26	11	67	427,166	5,618
DMU 8	Simple Python Keylogger for Windows	1	1	176	1,730	2,428
DMU 9	OSSIM	6	21	181	1,603	3,868
DMU 10	Umit	10	7	201	736	8,424
DMU 11	Kerberos Module for Apache	2	2	274	498	64
DMU 12	Pop Top	1	10	278	10,832	158
DMU 13	JGuard	10	16	300	458	19,651
DMU 14	Pam_mount module	5	3	302	502	228
DMU 15	Shellter	5	1	303	11,161	50,318
DMU 16	Another File Integrity Checker	0	3	322	433	488
DMU 17	BASE	12	11	326	4,824	311
DMU 18	GnomeSSH Tunnel Manager	5	1	356	1,074	184
DMU 19	Network Security Toolkit	1	3	410	10,007	269,811
DMU 20	Security and Privacy Complete 2	1	1	415	4,601	154
DMU 21	J2EE Certificate Authority, EJBCA	0	12	433	888	12,613
DMU 22	Dariks Boot and Nuke	7	5	459	78,186	2,082
DMU 23	Inprotect	5	7	459	738	651

DMU 24	Proxymizer	1	5	468	503	572
DMU 25	Grid Programming Environment	13	34	524	91	6,538
DMU 26	SNARE – Auditing and EventLog Management	4	10	534	7,153	1,160
DMU 27	AWStats	82	5	539	30,131	1,049
DMU 28	BlockSSHD	0	2	580	235	11
DMU 29	AxCrypt	13	2	618	24,681	1,041
DMU 30	MailArchiva	1	1	660	288	13,194
DMU 31	OpenCA	7	20	772	1,210	6,860
DMU 32	PhpRADmin	2	2	894	810	106,667
DMU 33	OpenSIMS	2	6	1,641	492	159,959
DMU 34	ClamMail	4	1	1,719	1,507	863

The "labor" that goes into an OSS project can be viewed as the persons contributing to the project. Contributors to an OSS project include software developers and bug submitters. These numbers are readily available through sourceforge.net. Thus, this research study includes both the number of developers and the number of bug submitters as input variables.

In this research, "kilobytes per download" is used as a measure of "useful" project size that naturally corrects for bloated code that is of little use to the end-user. The "number of downloads" indicates the level of acceptance of the software by the user community. "Project rank" is assessed directly by Sourceforge.net based on traffic, communication, and development statistics collected by Sourceforge.net for each project. The CCR, BCC, and SBM models have been used to evaluate the projects and results are presented in table 2.

Table 2: The obtained Performance results using Data envelopment analysis methods (CCR, BCC, and SBM)

DMU	θ_{CCR}	θ_{BCC}	θ_{SBM}
DMU 1	0.261	0.14	0.0014
DMU 2	0.163	0.10	0.0030
DMU 3	0.353	0.13	0.0090
DMU 4	0.059	0.03	0.0002
DMU 5	1.00	0.51	0.0486
DMU 6	0.095	0.03	0.0033
DMU 7	1.00	0.12	0.0026
DMU 8	1.00	1.00	0.0104
DMU 9	0.048	0.07	0.0005
DMU 10	0.142	0.27	0.0089
DMU 11	0.468	0.71	0.3240
DMU 12	0.132	0.20	0.0102
DMU 13	0.059	0.10	0.0105
DMU 14	0.306	0.51	0.2999
DMU 15	1.00	1.00	0.0464
DMU 16	1.00	1.00	1.0000
DMU 17	0.085	0.11	0.0031
DMU 18	0.883	1.00	0.0061
DMU 19	1.00	0.60	0.0337
DMU 20	0.929	1.00	1.0000
DMU 21	1.00	1.00	1.0000
DMU 22	0.508	0.20	0.0031
DMU 23	0.117	1.00	0.2893
DMU 24	0.162	1.00	0.3667
DMU 25	0.023	0.04	0.0017
DMU 26	0.082	0.14	0.0009
DMU 27	0.266	0.20	0.0005
DMU 28	1.00	1.00	1.0000

DMU 29	0.579	0.50	0.0023
DMU 30	0.756	1.00	1.0000
DMU 31	0.031	0.17	0.0005
DMU 32	0.651	1.00	1.0000
DMU 33	0.296	0.37	0.0015
DMU 34	0.045	1.00	1.0000

The first column of Table 2 records the efficiency measure by CCR model (model 1). The results show that eight projects out of 34 OSS projects based on security have the efficiency measure equal to one (efficient units).

According to the second columns in Table 2, the second column of BCC model (model 2), the efficiency of 12 projects (8, 15, 16, 18, 20, 21, 23, 24, 28, 30, 32 and 34) of the 34 open source software projects is equal to one and considered as efficient projects. Other projects with a performance score of less than one are considered as inefficient projects. It is difficult to rank efficient projects with equal performance based on the BCC model.

Also, according to performance column of SBM model in Table 2 (fourth column), the performance score of 7 projects (16, 20, 21, 28, 30, 32 and 34) of the 34 open source software projects is equal to one and again it is difficult to rank these efficient projects with the same performance score. Other projects with a performance score of less than one are considered as inefficient projects and all of them are ranked according to their performance score. Because of the weakness of CCR, BCC and SBM models for ranking all projects, therefore the super-efficiency model (model 4) is used to rank all projects. AP-super efficiency model (model 4) is one of the most popular and widely used ranking methods in the DEA literature. In order to have more discrimination of efficient units super-efficiency (AP) model (model 4) has been used on the data set and the results are recorded in Table 3.

As Table 3 reports, the super efficiency score of each project is unique. Applying the AP super-efficiency model (model 4), projects with score of more than 1 are called efficient units, and in this practical example, the efficient projects are: #16, #20, # 28, #30, # 32, and #34. Also, the projects with super efficiency score are less than 1, is called inefficient projects and their number is 28 projects and can be seen in table 3. The last unit (unit#34) is ranked as a first unit and with reference to their super efficiency score, unit#16 is placed in the second position. According to efficiency numbers all units are ranked. The last column of Table (3) demonstrate the unit ranking.

Table3: Ranking 34 Open source software projects with AP super efficiency model

Project Number	Super efficiency score	Ranking
1	0.0189	32
2	0.0191	31
3	0.1231	23
4	0.003	34
5	0.4101	16
6	0.0081	33
7	0.4833	13
7	0.0959	26
8	0.2667	18
9	0.0218	30

10	0.2596	19
11	0.6148	9
12	0.1381	22
13	0.0932	27
14	0.4755	14
15	0.5102	12
16	3.0145	2
17	0.1123	25
18	0.7438	7
19	0.6009	10
20	1.0091	6
21	0.5127	11
22	0.1391	21
24	0.6582	8
25	0.0357	29
26	0.1219	24
27	0.1633	20
28	2.7019	3
29	0.4682	15
30	1.4765	4
31	0.092	28
32	1.4713	5
33	0.3546	17
34	4.8016	1

RESULTS

This research evaluates the relative performance of security-based OSS projects by evaluating how efficiently numerous project inputs produce numerous project outputs. The number of unique users who have reported software defects and the total number of developers for a project are the two inputs examined for the 34 projects. The outputs for each project include the project's Sourceforge.net rank (lower values signify higher ranks), the number of Sourceforge.net downloads, and the amount of Kilobytes per download.

In OSS development, feedback from developers and bug contributors is critical for understanding overall project success. For the first time in this research, the authors used BCC, SBM, and AP-super efficiency models to evaluate the performance of Open source software projects and obtained performance value using GAMS software. As illustrated in the table 2, by using of the BCC, CCR and SBM models and couldn't rank all units, so chose AP- super efficiency model (one of the most used rating methods in DEA), and were able to find a different rating for all open source projects.

It's fascinating to compare the DEA model rating results to the SourceForge project rankings. One could assume that a DEA "efficient project" would inevitably be a "highly ranked" SourceForge project because both DEA and SourceForge take many of the same aspects into account. The DEA model, on the other hand, is simply concerned with how effectively each project delivers its results. The SourceForge rating is an activity rating that does not take efficiency into account when deciding project rank. This can be shown by examining the three projects that

have best rank, "ClamMail", "Another File Integrity Checker" and "BlockSSHD". These three projects were not at the top of the SourceForge ranking. In fact, Table 1 shows 34 security-based projects have a Sourceforge ranking higher than ClamMail, 16 security-based projects ranked higher than Another File Integrity Checker and 28 security-based projects ranked higher than BlockSSHD. In fact, none of the top five SourcForge ranked security-based projects was selected as an efficient project by the DEA model that was used in this research.

The findings of this research may be used by an OSS project manager to objectively assess resources for their project and then judge their relative performance. Also, based on the results of the DEA, managers can decide whether to increase or decrease resources, and effective strategies to achieve these objectives can be created. For example, the need for more people reporting software bugs may prompt the OSS project manager to adopt a new bug reporting tool to increase to social network among bug. As another example, DEA results may suggest a reduction on the number of developers. Ultimately, DEA has been shown to be a practical tool for ranking OSS projects.

REFERENCES

Andersen, P., & Petersen, N. C. (1993). A procedure for ranking efficient units in data envelopment analysis. *Management science*, *39*(10), 1261-1264.

Banker, R.D., Charnes, A., & Cooper, W. W. (1984). Models for the estimation of technical and scale efficiencies in data envelopment analysis. *Management Science*, 30(9), 1078-1092.

Charnes, A., Cooper, W. W., & Rhodes, E. (1978). Measuring the efficiency of decision making units. *European journal of operational research*, 2(6), 429-444.

Crowston, K., & Shamshurin I. (2017). Core-periphery communication and the success of free/libre open source software projects, *Journal of Internet Services and Applications*, 8(10), 1-11.

Fitzgerald, B., (2018). Crowdsourcing software development. *Software engineering und software management 2018. Bonn: Gesellschaft für Informatik*. (S. 23-24).

Flitman, A. (2003). Towards meaningful benchmarking of software development team productivity, *Benchmarking: An International Journal*, 10(4), 382-99.

Gao N., Richard S. S., Zichen Z., & Zhijian W., (2021). A Survey of open source statistical software (OSSS) and their data processing functionalities, *International Journal of Open Source Software and Processes*, 12(1), 1-20.

Hosseinzadeh Lotfi, F., Jahanshahloo, G. R., Khodabakhshi, M., Rostami-Malkhalifeh, M., Moghaddas, Z., & Vaez-Ghasemi, M. (2013). A review of ranking models in data envelopment analysis, *Journal of Applied Mathematics*, 2013, 1-20.

Jorgensen, N. (2001). Putting it all in the trunk: incremental software development in the FreeBSD open source project, *Information Systems Journal*, 11(4), 321-336.

Kalina, I. & Czyzycki, A. (2005). The ins and outs of open source, *Consulting to Management*, 16(3), 41-7.

Lacy, S. (2005). Open source: now it's an ecosystem, *Business Week Online*, October, p. 3.

Monteiro, N. S., Fontinele, A. C., Campelo, D., & Soares Mookhey A. (2020). Provision of adaptive guard band in elastic optical networks", *Journal of Internet Services and Applications* , 11(5), 1-19.

Paradi, J.C., Reese, D.N., & Rosen, D. (1997). Applications of DEA to measure the efficiency of software production at two large Canadian banks, *Annals of Operations Research*, 73, 91-115.

Payne, C. (2002). On the security of open source software, *Information Systems Journal*, 12(1), 61-78.

Pereira, J. (2021). Leveraging final degree projects for open source software contributions, *Electronics*, 10(10), 1-16.

Heidary, S., Zanboori, E., & Parvin, H. (2018). A Hybrid model based on neural network and data envelopment analysis model for evaluation of unit performance", *Iranian Journal of Optimization*, 10(2): 2, 101-112.

Stensrud, E. & Myrtveit, I. (2003). Identifying high performance ERP projects, *IEEE Transactions on Software Engineering*, 29(5), 398-415.

Yang, Z. & paradi, J.C. (2004). DEA evaluation of Y2K software retrofit program, *IEEE Transactions on Engineering Management*, 51(3), 279-287.

Zanboori, E., Hosseinzadeh Lotfi, F., Rostamy-Malkhalifeh M. & Jahanshahloo, G. R. (2014). Calculating Super Efficiency of DMUs for Ranking Units in Data Envelopment Analysis Based on SBM Model. *The Scientific World Journal*, 2014, 1-7.