How Healthy Is My Project? Open Source Project Attributes as Indicators of Success

James Piggot, Chintan Amrit

How Healthy is my Project? Open Source Project Attributes as Indicators of Success

James Piggot[1] and Chintan Amrit[2]
Department of IEBIS
University of Twente
The Netherlands
1. j.j.h.piggott@student.utwente.nl
2. c.amrit@utwente.nl

Abstract. Determining what factors can influence the successful outcome of a software project has been labeled by many scholars and software engineers as a difficult problem. In this paper we use machine learning to create a model that can determine the stage a software project has obtained with some accuracy. Our model uses 8 Open Source project metrics to determine the stage a project is in. We validate our model using two performance measures; the exact success rate of classifying an Open Source Software project and the success rate over an interval of one stage of its actual performance using different scales of our dependent variable. In all cases we obtain an accuracy of above 70% with one away classification (a classification which is away by one) and about 40% accuracy with an exact classification. We also determine the factors (according to one classifier) that uses only eight variables among all the variables available in SourceForge, that determine the health of an OSS project.

1 Introduction

Determining what makes a software project successful has been a research topic for well over 20 years. The first model that defined the factors influencing software success was published in 1992 by Delone and McLean [1], as the Information Systems Success Model. Since then there has been a considerable effort in research to determine what can be done to minimize project failure. However, factors that influence commercial projects differ from those known as FLOSS or Free/Libre Open Source Software. Attempts at remedying this gap have focused on statistical models that focus on certain aspects of a software development lifecycle. Only recently, has historical data been used to determine the changing nature of factors for success during a projects lifecycle[2].

In this paper we use machine learning in the form of decision trees, to predict the development stage of an Open Source project based on project metrics[1], project

[1] We use the terms metric and attribute to mean the same concept in this paper

constraints and circumstance. This model will serve as an indicator of OSS project health that will enable developers to determine accurately in what stage their project is in and what is necessary to improve project success. For organizations seeking to use OSS it can also be used to determine what risks are associated with sponsoring a project.

Previous research has tried to understand which indicators influence a project's success and how these indicators are interrelated but there have been very few working models[3]. What this paper proposes is that, through machine learning we can model which available project metrics are of importance in determining OSS project health. Our method differs from previous attempts at building a model, which were based on statistical correlations that approximated success factors without revealing how they actually influenced project's status [4-6].

In the last few years a considerable number of papers have been published that have tried to determine what the indicators of OSS project success are and how these indicators are interrelated. Often a number of these metrics are empirically tested on OSS projects found on SourceForge[2], a key OSS depository. In this paper we try to estimate the status of a project based on various metrics related to an OSS project. We also determine the accuracy of the subjective status of an OSS project in SourceForge that is provided by the OSS project leader. To this extent, we extend the research work on the problems with reporting the status of a software project [7], to OSS projects. For the purposes of this research we use SourceForge to obtain a data sample and use longitudinal data collected from 2006 to 2009. We have limited the sample to projects starting in 2005, in order to observe all stages of a project's lifecycle.

2 Literature Review

Recent research on OSS success factors has focused on enlarging the scope of influences, common elements found cite factors such as user/developer interest [2], the critical number of active developers [8] and software quality [5].

Ever since the publication of the Information Systems Success model by Delone and Mclean[1] researchers have attempted to define in what way factors that influence Open Source Software differ from those of commercial software. Early research showed that due to geographical dispersal of developers and lack of formal managerial methods, coordination becomes more difficult [9], this has been off-set by the proliferation of software forges that act as a single locale for communication and development for a project as well as a download site for users. To determine which metrics found on a software forge can be used to determine project success, an explanation of the IS success model is in order.

[2] http://www.sourceforge.net

2.2 Open Source Health

In order to gauge the success of the Open Source projects we studied in this paper, we looked into literature on measuring Open Source success.

Crowston et al. [3] collect data on the bug tracker and the mailing list of the projects to determine the health of the projects. They propose that the structure of the OSS community determines the health of the community and state that an onion structure is one of the better OSS community structures. Subramanian et al. [2] measure an Open Source project's success by measuring user interest, project interest and developer interest. They measure user interest by calculating the number of project downloads and to measure the developer interest in the project, Subramanian et al. [2] count the number of active developers in the project. Finally, they measure project activity by calculating the number of files released in the project [2].

Other authors such as Stewart et al. [10] find that licence choice (i.e. how restrictive the licence is) and organizational sponsorship (i.e its affiliation with a for-profit company or university) determine how successful the OSS projects are. In addition to these measures, Sen et al. [11] also find subscriber base (i.e. the number of individuals who chose to be updated about the project developments) and number of developers to reflect the "healthiness" of an OSS project [11]. Chenglur-Smith et al.[12] work on similar lines, and predict that a OSS project's age and size help in the sustainability of the project (i.e. its ability to retain interest and continue to attract developers) [12]. Amrit and Hillegersberg [13], on the other hand, explore the core-periphery shifts of development activity and its impact on OSS project health. They find that a steady movement of developers away from the core of the software code is indicative of an unhealthy OSS project [13].

Regarding the techniques used to analyse OSS data, English and Schweik [14] produce a six-part classification for OSS projects. They base this classification on phone interviews with OSS developers, manual coding of a sample of OSS projects from SourceForge.net, and theoretical insights from Hardin's "Tragedy of the Commons". English and Schweik operationalize these definitions and test them on 110,933 SourceForge projects, with low error rates. Wiggins and Crowston [15] extend this research and analyse another SourceForge data set. Of 117,733 projects, they classify 31% as abandoned at the Initiation stage, 28% as abandoned at the Growth stage, and 14% as successful at both the Initiation and Growth stages.

Though the dependent variables of English and Schweik [14] are well thought out, they do not explore the relationship of their classification with the existing classification of projects in SourceForge. Furthermore, their focus is by and large to determine the number of successful and unsuccessful OSS projects and to classify projects into their six categories. In this paper, we also try to determine the factors that affect the heath of an OSS project. Specifically, we try to predict the subjective classification provided by the project managers and developers of the different SourceForge projects in order to (1) check the validity of the subjective classification and (2) if the classification is indeed valid, one can use the classifier to determine the variables that affect project health.

2.3 Success Factors

Previous research has focused on using three well known metrics to determine project success with the added benefit they have corresponding metrics on SourceForge [2, 3].

The use of longitudinal data from past projects hosted by SourceForge.net to determine OSS success is also an innovation in recent studies [2]. They divide the independent variables into two groups; time-variant and time-invariant and determined how they affect the success measures. The outcome of this study validates the idea of using historical data, as they have proved that past levels of developer and user interest influence present interest. The effect of this change in popularity is that lead-developers and project managers should better anticipate the future need for resources and manage both the internal and external network size of a project [16].

The choice of software license can also have a detrimental influence on the success of a project [4]. They find that if more effort is necessary to complete the project, developers tend to choose less restrictive licenses such as those from the No-Copyleft or Weak-Copyleft categories as opposed to Strong-Copyleft. This even holds true when developers prefer to use more restrictive licenses to ensure that derivative work is adequately protected. The choice of license can be influenced by external factors such as royalties and network effects. As such the preferred license can differ from the optimal license. Other research has shown that when a project has managed to pass through the initial stages of its lifecycle with a less than optimal license it will not severely influence future success [17].

A difficult topic to study relates to determining what factors influence OSS projects in both the initial stage and in the growth stage. As data from a project's initial stages is often absent, this resolves to determining what time-invariant factors can influence the growth stage. Research has found that the initial stage of an OSS project is indeed the most vulnerable time period as a project competes for legitimacy with other similar projects in attracting developers.

3 Methodology

3.1 Data and variable definitions

In order to build a model that approximates the status value of a software project on SourceForge we need to gather factors that might influence developers to assign a particular value of status.

Table 1: Overview of SourceForge's status classification

Classification Number	Development Stage
1	Planning
2	Pre-Alpha
3	Alpha
4	Beta
5	Production / Stable.
6	Mature

7	Inactive

3.2 Dependent variables

To determine the success of software project we try and categorize what status (stage) of development a project has reached.

3.2.1 Project Status

The current progress of a software project can be placed in one of five development stages according to the System development life cycle: these are Requirements Planning, Analysis, Design, Development and Maintenance [2]. Another way to describe a projects progress is through terms such as Planning, Alpha, Beta and Stable. Shifts in progress are marked by improvements in completeness, consistency, testability, usability and reliability [2].

SourceForge.net maintains a system of 7 status designations (Table 1). The numbers 1 through 6 are for Planning, Pre-Alpha, Alpha, Beta, Production/Stable and Mature The last status is an outside category for projects that are Inactive. Previous research [2] makes it clear that it can be expected that projects reaching advanced stages of their life cycle will be more in favor with users and those in earlier stages, as their input goes beyond mere maintenance. As more users make use of the software, they also generate more bug reports, feature requests and feedback/suggestions. In turn developers develop more patches. As such, the latter stages of development are marked by more development activity related to patches, bugs and feature requests. The use of historical data in previous research [2] shows that increased numbers of developers and users will show later on in increased project activity. To better mark this relationship between different time periods within a project we have selected projects on SourceForge that have valid data from a period of four years from 2006 to 2009. The 7 stage status category used by SourceForge is considered by some [2, 11], an awkward use of the typical lifecycle definitions used in software development. SourceForge uses only vague descriptions for each, and much is left to developers to decide the status their project is in. Especially the difference between pre-alpha and alpha, as well as between production/stable and mature, may be cause for confusion. To overcome this, we also consider the binary project status representing whether the project is active, and the project status variable that has four categories; namely Planning, Alpha and Beta and Stable. To achieve the four stages of the project status, we collapsed the inactive and planning stages to the Planning stage, we aggregated the stages Pre-Alpha, Alpha to Alpha stage and Production/Stable and Mature to the Mature category.

Projects on SourceForge.net can also be differentiated along a different dimension, that of project activity. It can be reasonably assumed that both projects in the planning stages and those that are inactive do not have either code to download or developers to work on the project. As such they should be markedly different from those projects of which code is available for download and alteration. We propose to

check for ways in which projects that are inactive differ from those that are active, apart from the aforementioned variables.

3.3 Time-invariant variables

Variables that can influence the success of an OSS project can be divided into two groups – time-invariant factors and time-variant factors. The variables included have been previously identified in literature as affecting OSS success [3, 11].

For time-invariant variables we have chosen those that define a project in general terms such as license [4], the operating system that can be used and the programming language [2] in which the code is written, to determine if they are factors that have an influence on the project status. Each variable is divided into binary variable categories such as Strong-CopyLeft, Weak-Copyleft and No-Copyleft for license and after which each project is assigned either the value 0 or 1 to show if it supports a particular feature.

The time-invariant variables have been further augmented with simple numerical variables that list the number of features of each category that a project supports. So for license, there is a variable that would count the number of licenses used by a project.

License

The license used by a project can influence the amount of support it gets, as it affects the interests of users and developers [10]. Software licenses can be broadly divided into three groups based on the level of restrictiveness that would determine whether users can distribute derivatives or modify the software (copyfree).

These categories are Strong-Copyleft, Weak-CopyLeft and No-CopyLeft. Licenses such as GPL (General Public License) and BSD (Berkeley Software Distribution) License are grouped into these categories depending on whether they support issues such as ‘copyfree’ or not. Various research papers already use this division of licenses and where licenses fall into [9, 10]. However, numerous licenses cannot be exactly assigned to any of the three categories because they do not conform the GPL format. As far as possible, they are assigned based on the effect these licenses have on user and developer choices.

Operating System

The operating system used for development and use of a project can have a severe impact on its popularity as it determines how many users it could potentially reach as well as what type of license the developer intends to use [2]. Traditionally open source software has used UNIX, Linux and its derivatives for development, which caused OSS to be somewhat excluded from other operating systems such as Windows and Mac OS X. With the popularity of languages such as a Java that make portability possible Windows has become an increasingly more popular for OSS developers. Previous research[2] has indeed focused on these three categories of operating system: the Windows-family, UNIX (also includes Linux and POSIX) and a other category that includes MAC OS X. They prove that UNIX and Linux type operating system have a negative correlation with user interest, but a positive

correlation with developer interest and explain this based on the roots of the OSS community who frequently started their career on UNIX and Linux machines.

We have expanded the number of OS groupings to also include 'OS Independent' as a category to denote the increasing popularity of portability. Mac OS X has also been grouped into a separate category as an acknowledgement of its increasing popularity. Other operating systems were left out of this study. The increase in number of categories should allow for better rules to be deduced from our data mining efforts.

Similar to the license variables these categories denote binary variables and an outside category has been added that counts the number of operating systems supported by a project.

Programming Language

The effects of the programming language used in a project has previously solely focused on the C-family of languages, while others where either excluded from study or aggregated into one category [2, 3]. This study intends to rectify this deficiency to also include popular languages such as Java and PHP as separate categories without denying the continued importance of C-type languages.

Because the C programming language was used for the implementation of UNIX is has remained popular with UNIX and Linux developers ever since [2]. Despite memory allocation problems it has remained a favorite for projects that have more stringent processing and real-time requirements. Through the prevalence of high quality compilers and the importance of derivative languages (C++ C# and Visual C++) the use of C can be associated with more developers and project activity [2].

For our study we have expanded the number of language categories to 5 and included 'C-family', 'Java', 'PHP', 'Python' and 'Others' as separate categories.

3.4 Time-variant variables

The three success measures previously mentioned that have their roots in the IS Success Model also have their equivalents in OSS projects found on SourceForge. These are Project Activity (number of files, bug fixes and patches released), User Activity (number of downloads) and Developer Activity (number of developers per project). Crowston et al. (2006)[3] discovered that these measures are interrelated as developers are often users which means that the number of downloads and developers is thus correlated. Project Activity is also closely correlated with User Activity as the latter often download the latest software releases, developers tend to flock to such projects as well. We use the above three metrics as the basis for our time-variant variables.

Other variables include the number of donors, forum posts, mailings lists, feature requests and 'Service Requests' that allow users to ask for help from developers. Our dataset also includes the project age in days from 2009 backwards as a control variable. Combined, we have constructed a dataset that contains 38 variables including 35 independent variables and three variations of one dependent variable, i.e. project status with 7, 4 and 2 project statuses.

3.5 Dataset sampling

SourceForge.net is the largest web portal for the development of Open Source Software, it acts as a repository for code, as a tracking system for bugs and features and as a communication outlet for those involved in software development. As of November 2012 it hosts some 300,000 projects that differ in a wide range of categories such as intended audience, the topic of the project, the license used as well as technical attributes in which projects can distinguish itself; programming language, OS supported as well as the Graphical User Interface used. For the purpose of this study it is impossible to gather data directly from SourceForge through a screen scraper as the servers of Sourceforge.net cannot distinguish this activity from more nefarious ones such as a Denial of Service attack.

The dataset used has thus been obtained through a third source which has made the data publicly available [18]. FlossMole.org contains data collected for the period 2006 to December 2009 from which a dataset was compiled of 125,700 projects. Unfortunately, many projects had missing data, due to the fact that no data was entered by developers, or project portals were not maintained, or the screen scraper that collected data often did so wrongly, which corrupted portions of the dataset.

Our dataset initially contained 125,700 projects and most projects had incomplete data for the time period 2006 to 2009. Hence, upon cleaning the data we were left with 28,282 rows in our database,

4 Experiment Methodology

We used the SPSS 2.0 decision tree analysis able to analyze the data and predict project status. In our research we chose the CHAID[19] and the CART[20] method of data classification, in order to handle over 35 independent variables some of them being categorical, numeric and non-parametric.

Decision trees can suffer from over-training, whereby the trees continue to grow and might afterwards not be able to validate test-data because it uses rules learned from the training data that are incompatible with the test data. Both CHAID and CART use different ways to limit the growth of decision trees.

CHAID

Which stands for ‘Chi-squared Automatic Interaction Detection’ uses a statistical stopping rule to keep the tree from growing to impossible sizes. CHAID has the advantage of being able to handle categorical variables. Other research using this method indicate that it excels at analyzing all the factors that can possible influence a dependent variable but it’s result at predicting these value with subsequent data samples is often poor.

CART

Also known as ‘Classification and Regression Trees’ builds a tree based on theory quite different from CHAID. CART uses a non-parametric approach which can work with both categorical and numerical variables, and also has the ability to

model the complex interactions among the variables[20]. It first grows the tree to its full size and afterwards it prunes the tree until the accuracy of the tree is similar for both the training dataset and the test dataset.

The reason why we chose CHAID and CART is that while both classifiers can work with categorical variables and use different theoretical models, they are also comparable in some aspects (lift in response)[21].

Cross Validation

With both methods of growing a decision tree we have used our dataset in two ways. The first is cross-validation of the entire data sample whereby data is partitioned multiple times over in both trainings sets to build the tree and test sets to validate the tree.

This is a process whereby data is manually split into training and test sets. For the purposes of this study we used a 50-50 data split in order to avoid overtraining.

5 Results

Below are the results of for each of the three types of project status. The results include both the CHAID tests as well as the Cross-validation and data-split methods.

The accuracy with which our classification tree has been able to determine the correct project status can be seen in tables 2 and 3.

5.1Data split

The results in table 2 have been obtained through a 50-50 data split (to prevent overtraining) and represents the results of the training set. The method obtained two results of importance, the first is the exact match of 39.6 % whereby of the 24582 data samples 9729 achieved the correct corresponding status value. For a 7-fold category this result can be considered acceptable (as compared to 1/7 ~ 0.14 for random chance). The second method, or 1-away result, shows what percentage of data samples either had exactly the correct status value, or were just 1 value off the mark. The accuracy for this is 76.2 % and suggests that the results are closely distributed around the correct value. This result validates the decision tree that was grown from the rules deduced from this test.

Table 2 Results of the CHAID decision tree with 7 stage project status

	Categories.							Accuracy.	
	1	2	3	4	5	6	7	Exact match	1-away
1	3246	539	115	344	113	0	27	74.0%	86.3%
2	1770	809	189	1029	257	0	16	19.9%	68.0%
3	967	368	400	2083	534	0	10	9.2%	69.2%
4	786	262	279	3469	1175	0	9	58.0%	78.8%
5	386	109	213	2478	1734	0	3	35.2%	85.4%
6	39	14	10	184	132	0	0	0.0%	34.8%
7	180	34	32	120	47	0	71	14.7%	14.7%
								39.6%	
									76.2%

Exact match = 39.6%.
number of hits / total cases = 9729 / 24582.
1-away match = 76.2%.
(number of hits + number of 1-away hits) / total cases
= (9729 + 9013) / 24582.

5.2 Cross-validation

For the cross-validation method, the final score seems to closely match those of the data split method. However, for status value corresponding to Planning (1) and Inactive (7) the results differ significantly, as this method partitions the data set multiple times it would average out the more extreme values obtained through the data-split method.

Table 3: Status results through cross-validation.

	Categories.							Accuracy.	
	1	2	3	4	5	6	7	Exact match	1-away
1	6008	1352	288	691	181	0	43	70.2%	85.9%
2	3043	2084	590	1913	471	0	8	25.7%	70.5%
3	1711	924	1364	3551	1230	0	15	15.5%	66.3%
4	1420	664	910	6693	2328	0	27	55.6%	82.5%
5	666	386	718	4321	3710	0	5	37.8%	81.9%
6	72	45	70	283	275	0	0	0.0%	36.9%
7	358	76	73	236	93	0	70	7.7%	7.7%
								40.7%	
									76.0%

Exact match = 40.7%.
Number of hits / total cases = 19929 / 48966.
1-away match = 76.0%
(number of hits + number of 1-away hits) / total cases
= (19929 + 17294) / 48966.

Both results validate our method to classify software projects found in SourceForge.

5.3 Four stage project status: Planning, Alpha, Beta and Stable

In figure 4 are the results of our efforts to classify projects in the four categories popularly described in literature. The test results were obtained using the CHAID method with a 50-50 split of the dataset.

Table 4 Results of CHAID for 4 stage project status

	Categories.				Accuracy.	
	1.	2.	3.	4.	Exact match	1-away
.	1877	2081	157	73	44.8%	
.	1270	5049	1274	858	59.4%	
.	424	2405	1837	1327	30.7%	
.	201	1621	1384	2094	39.5%	
					45.4%	
						86.0%

Figure 1. Stage results.

Growing method; CHAID.
Exact match = 45.4%.
Number of hits / total cases = 10857 / 23932.
1-away match = 86.0%.
(number of hits + number of 1-away hits) / total cases
= 20598 / 23932.

The accuracy of 45.4% is better than the score for the 7-fold status category, though it's predictive value is especially undermined by the low score in its efforts to classify projects in Beta stage. This could be seen as proof that this stage is a subjective stage that is hard to classify through machine learning. The 1-away score of 86.0 % once again proof that the scores are distributed around the correct value though for a 4-fold category the value loses in importance.

This result validates that out method works to determine the correct stage in its lifecycle a software project is in.

5.5 Binary Project Status: Active and Inactive

We get an accuracy of 82.4% for classification of projects based on a binary status of active or inactive. The results can be considered to be better, if we consider that the inactive state is an aggregation of the Planning status and the Inactive status, they share many things in common but also have crucial differences for the former can have developers assigned to it.

Table 5, Results of CHAID for a binary project status

	Categories.		Accuracy
	1	2	
1	1226	3539	25.7%
2	789	18985	96.0%
			82.4%

5.6 Cross validation result.

The cross-validation method seems better able to determine whether a project is active (1), because the method splits the dataset 10 times and tests each iteration we can presume that the lower score for the above 50-50 data split is an aberration.

Table 6: Results of CHAID, CV for binary project status

	Categories.		Accuracy
	1	2	
1	3498	6972	36.9%
2	2474	37023	93.7%
			82.8%

This result of 82.8% accuracy, shows that our method can successfully distinguish active projects from inactive projects.

6. Discussion

The results of our classification show a nearly 40% accuracy for an exact match of the subjective classification and a 76% 1-away match (Table 2) indicates that subjective classification performed by the OSS project leader is quite accurate and correlates with the project data. This is quite unlike what is reported for commercial projects [7]. The errors and implications of this finding can be a subject for future research.

By using the CART method of decision tree analysis, we obtained a model for status classification, as shown in Figure 3. The tree shows that numerical metrics such as downloads, donors, developers and forum posts are far more explanatory of

project health than time-invariant metrics such as license used, or the operating system supported.

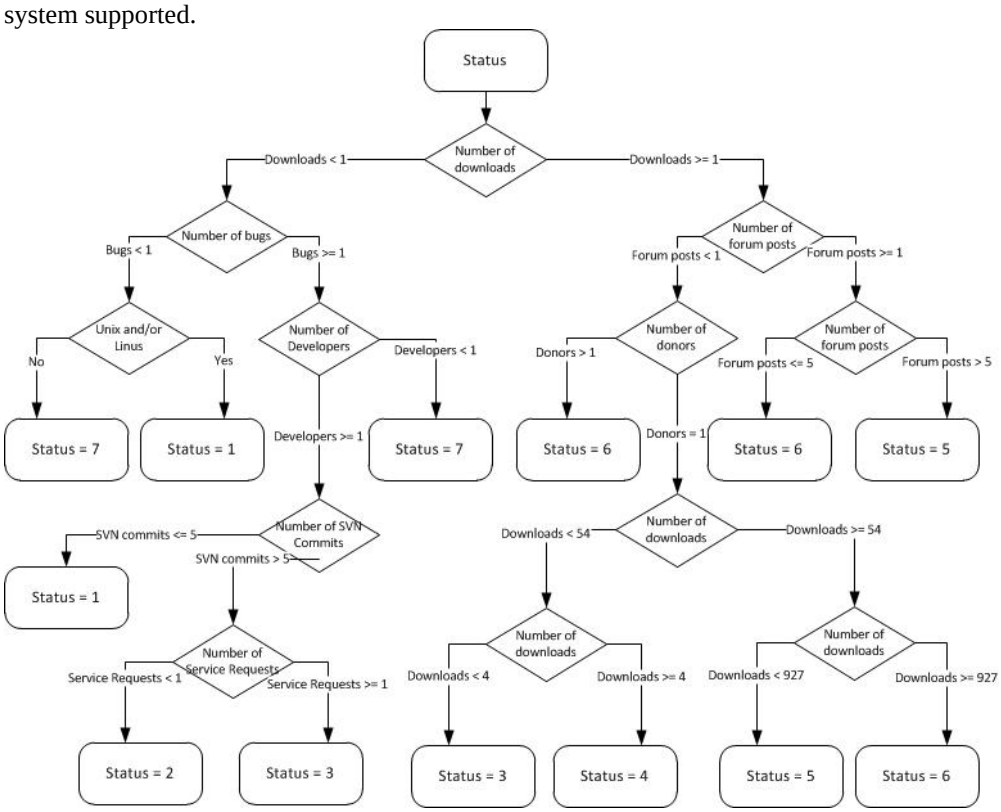

Figure 2: The CART decision tree for our data

This is in-line with earlier research [2]. However, this should not be surprising as those time-invariant metrics are usually decided upon when the project is initiated, and change little over its lifecycle. When they do change, they change only to suit users and developers. On the other hand, time-variant metrics, by their definition, can gauge what popularity a project has presently obtained. The order of importance that metrics have taken in the model is also as expected and follows established literature [2, 3].

In the early stages of a project lifecycle, the ability to attract developers is of vital importance in order to be able to develop software along the stated goals. In the latter stages of the software life cycle, it can be expected that users and developers generate more forum posts. This model also shows that the number of donors is an indication as to whether the project has status 4, 5 or 6. This can serve as an indicator, for example, that a project sponsor can have a positive influence on project success. This is in line with the findings of Stewart et al.[10].

The number of SVN commits relates to the number of changes developers have uploaded to the central software repository on SourceForge. In the model (Figure 3)

it is closely related with the number of developers on a project in the early stages of the software lifecycle. The number of CVS commits denotes the number of official software releases and surprisingly is not part of the model obtained.

The choice of license is also not an important factor in determining whether a project will be able to continue to succeed in the growth stage. This validates other research [10, 17] but for the most part contradicts long established views on how a project would compete for resources.

Even though our dependent variable is the SourceForge subjective classification done by the OSS project leaders, we can definitely say that given the predictive accuracy of 1-away classification, the classification model does reflect the stage/health of the OSS projects. As validation of this claim, we find that most of the important variables are also mentioned by other authors[2, 3, 5, 6]

With our model we have managed to predict the status of a project with reasonable accuracy. The model in figure 3 shows this when status 7 (inactive) is reached after a combination of few downloads few active developers and only a small number of bug reports have been generated.

7. Conclusion

We make two primary contributions with this research: (i) we demonstrate that the subjective project status (especially the 1-away value) reflects the actual health of the OSS project. This finding is in line with that of [2] and shows that, in this respect OSS projects differ from commercial projects [7] (ii) we determine the variables that affect project status and in turn affect project health based on nearly 30 K projects over a period of four years.

Our research shows that with a limited set of just 8 variables (Figure 3), we can gauge the status of a software project on SourceForge. Analyzing these 8 attributes of the OSS project can help alert Project controllers that their project is either poorly supported or will become obsolescent in the near future due to lack of developer interest. For prospective developers and sponsors this model can give an idea, whether a project is on track to pass through the early difficult stages of a software life cycle on schedule and is in fact not already failing.

We think our results can provide further research opportunities in projects that also suffer from users and developers being flooded with data, whose accuracy cannot be interpreted easily. Crowd funding sites such as Kickstarter[3] offers an index of project that are considered ‘popular’ and ‘most funded’ but there may be lopsided metrics as projects size, ambition and accessibility can negatively influence them.

REFERENCES

[1] W. H. DeLone and E. R. McLean, "The DeLone and McLean model of information systems success: a ten-year update," *Journal of Management Information Systems*, vol. 19, pp. 9-30, Spr 2003.

[2] C. Subramaniam, *et al.*, "Determinants of open source software project success: A longitudinal study," *Decision Support Systems*, vol. 46, pp. 576-585, 2009.

[3] http://www.kickstarter.com/

[3] K. Crowston, *et al.*, "Information systems success in free and open source software development: theory and measures," *Software Process Improvement and Practice*, vol. 11, pp. 123-148, 2006.

[4] S. Comino, *et al.*, "From planning to mature: On the success of open source projects," *Research Policy*, vol. 36, pp. 1575-1586, 2007.

[5] S. Y. T. Lee, *et al.*, "Measuring open source software success," *Omega*, vol. 37, pp. 426-438, 2009.

[6] V. Midha and P. Palvia, "Factors affecting the success of Open Source Software," *Journal of Systems and Software*, 2011.

[7] A. P. Snow and M. Keil, "The challenge of accurate software project status reporting: a two-stage model incorporating status errors and reporting bias," *Engineering Management, IEEE Transactions on*, vol. 49, pp. 491-504, 2002.

[8] A. Mockus, *et al.*, "Two Case Studies of Open Source Software Development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, pp. 309-346, 2002.

[9] J. Wang, "Survival factors for Free Open Source Software projects: A multi-stage perspective," *European Management Journal*, 2012.

[10] K. J. Stewart, *et al.*, "Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects," *Information Systems Research*, vol. 17, pp. 126-144, 2006.

[11] R. Sen, *et al.*, "Open source software licenses: Strong-copyleft, non-copyleft, or somewhere in between?," *Decision Support Systems*, 2011.

[12] I. Chengalur-Smith, *et al.*, "Sustainability of free/libre open source projects: A longitudinal study," *Journal of the Association for Information Systems*, vol. 11, p. 5, 2010.

[13] C. Amrit and J. van Hillegersberg, "Exploring the impact of socio-technical core-periphery structures in open source software development," *Journal of Information Technology*, vol. 25, pp. 216-229, 2010.

[14] R. English and C. Schweik, "Identifying success and abandonment of FLOSS commons: A classification of Sourceforge. net projects," *Upgrade: The European Journal for the Informatics Professional VIII*, vol. 6, 2007.

[15] A. Wiggins and K. Crowston, "Reclassifying success and tragedy in FLOSS projects," *Open Source Software: New Horizons*, pp. 294-307, 2010.

[16] R. Sharda and D. Delen, "Predicting box-office success of motion pictures with neural networks," *Expert Systems with Applications*, vol. 30, pp. 243-254, 2006.

[17] J. Wang, *et al.*, "Human agency, social networks, and FOSS project success," *Journal of Business Research*, 2011.

[18] J. Howison, *et al.*, "FLOSSmole: A collaborative repository for FLOSS research data and analyses," *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 1, pp. 17-26, 2006.

[19] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *Applied statistics*, pp. 119-127, 1980.

[20] L. Breiman, *et al.*, *Classification and regression trees*: Chapman & Hall/CRC, 1984.

[21] D. Haughton and S. Oulabi, "Direct marketing modeling with CART and CHAID," *Journal of Interactive Marketing*, vol. 11, pp. 42-52, 1997.