Activity Mining in Open Source Software

Daniel Mack
University of Notre Dame
dmack@nd.edu

Nitesh V. Chawla
University of Notre Dame
nchawla@nd.edu

Greg Madey
University of Notre Dame
gmadey@nd.edu

Abstract

Open Source software repository is a collection of various projects with varying levels of activities, participations, and downloads. In this paper, we attempt to categorize and mine activity, thus discovering success, by focusing on the *Games* group. However, we observe that there are a significant number of projects within the Games category that are inactive or have no ranking assigned by Sourceforge. So, we first segmented our dataset based on just the ranking or activity assigned. We then constructed a set of characteristics, such as number of developers; number of posters; number of downloads, to identify associations, activity, and relationships in the smaller active set. We used regression rules and association rules to discover the associations and relationships.

Contact:
Prof. Nitesh Chawla
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Tel: 1-574-631-3637
Fax: 1-574-631-9260
Email: nchawla@cse.nd.edu

Key Words: SourceForge, Open Source Software(OSS), Knowledge Discovery in Databases(KDD) , Weka, Association Rules, A Priori

Acknowledgement: The results reported in this paper are based in part upon work supported by the National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829.

Activity Mining in Open Source Software

Daniel Mack, Nitesh V. Chawla and Greg Madey
Department of Computer Science and Engineering,
University of Notre Dame,
{dmack,nchawla,gmadey}@nd.edu

ABSTRACT
Open Source software repository is a collection of various projects with varying levels of activities, participations, and downloads. In this paper, we attempt to categorize and mine activity, thus discovering success, by focusing on the *Games* group. However, we observe that there are a significant number of projects within the Games category that are inactive or have no ranking assigned by Sourceforge. So, we first segmented our dataset based on just the ranking or activity assigned. We then constructed a set of characteristics, such as number of developers; number of posters; number of downloads, to identify associations, activity, and relationships in the smaller active set. We used regression rules and association rules to discover the associations and relationships.

Keywords
SourceForge, Open Source Software(OSS), Knowledge Discovery in Databases(KDD) , Weka, Association Rules, A Priori

1. INTRODUCTION
There are two forces moving software forward in this world and they are currently at odds with one another. On one side is the guard that consists of computer scientists who espouse the standards of Software Engineering. Software engineering is the idea that in order to build efficient, maintainable software, certain ideas about group interaction must be strictly adhered to. In the other camp is the idea of open software [1,2,3,4], created from around the world, in which virtually anyone can examine the source, make changes, and attempt to improve the software. This flies in open and sometimes hostile defiance of traditional software engineering. In what appeared to be the limiting factor for this open source software was infrastructure, and its cost, open source is more often than not, free, and infrastructure is not. Out of this vacuum, an open source database called Alexandria, Its professional name is SourceForge and it was born and organized to provide free space and some basic administrative tools to allow these projects to gather and organize and better challenge the norm. These groups produce useful software many people use on a daily basis, and learning what makes a group successful would be helpful for other companies examining new ways to be efficient and also to sociologists who examine group interaction.

However, for every successful group on SourceForge, there are ten who do not attempt to produce software but still exist because they registered with SourceForge. The key is finding the successful groups. Success can be hard to conceptualize in a metric to measure. However finding the right groups to measure is luckily easier to conceptualize. It would only make sense that groups in the database that are active are at least communities worth studying. Thus we focus on how to find active groups in SourceForge

Activity in SourceForge is the metric used by common users as well as the database itself for finding projects that are popular and thus usually successful. It turns out that SourceForge calculates activity as a combination of number of downloads and the number of times the summary page for the project has been viewed. This is calculated over two time periods. The first is on a weekly basis. The other period is “all time” starting with when the project was founded. While weekly measurements have interest, the focus on long term success would be better to investigate, in terms of understanding the growth of a projects community. The metric of rankings and the percentile are what SourceForge gives these projects. Although far from scientific, a basic assumption is that, the more active you are in SourceForge, the more successful the project is in general terms. Thus projects with activity are more successful than non active projects, but the relative success inside the active category is still unknown.

This paper will proceed to discuss the different aspects of the project. Section 2 will contain a definition and description of the problem at hand. Section 3 will talk about sources of data and the main paper used in initially guiding this project. In Section 4, the discussion will focus on the approach as well as which components of the KDD process were necessary and the problems encountered. In Section five, the results will be discussed and analyzed. The last Section will deal with conclusions and any future work.

2. PROBLEM DEFINITON
Just by browsing SourceForge.net one can qualitatively see that the open source movement is large with a sizeable amount of projects and an increasing number of developers working on multiple projects [1,2,3,4]. Among the reasons that seem to encourage the increase in the numbers of groups registering, is that the amount of administration that SourceForge takes care of for the group and therefore enables the groups before they even begin. An infrastructure pre-built to allow groups to focus on the code certainly helps. Also making registration free for all brings a democratic feeling to software development.

This is both a blessing and a curse for researchers, while there are significant numbers of cases to analyze, the number of active groups is clearly outweighed by groups with good intentions but that never get off the ground. The need to isolate groups based on their relative activity is becoming increasingly important. The problem is this; can we data mine SourceForge to isolate metrics that will identify a variety of active projects?

Data mining is a necessity because the size of the community is so large and still growing, that a qualitative analysis such as browsing is no longer an efficient method. With this in mind, the problem of generating understanding about the activity of the open source community is one of dealing with a large number of projects, refining what relevant statistics can aid in the search, and which ways to relate these metrics to another and success.

3. SOURCES

3.1 Data

The raw data for this investigation will be in the form a Postgres database dump provided by SourceForge.net. The database will contain most of the information that is displayed on the web as well as information pertaining to statistics about users and the projects that they are currently involved in. No pre-processing has been done to the database other than removing information as it pertains to a user right to privacy. The data dump contains roughly 150 tables with varying amounts of attributes.

4. APPROACH

The approach was to find active groups and attributes that would aid in the search. To help facilitate the search, a limit was needed, in order to work with the data. This would be accomplished by concentrating on groups of a certain type of genre. The genre decided on was the *Games* category. Being a genre with a definite split (either activity or none at all) and relatively small (around 3802 projects) number of projects, it would be a good group to try and establish a model. The hope was that picking some features would easily divide the projects.

The following subsections will detail the algorithms used for the association rules and M5, and then secondly, the components that went into the project. Of course this paper is part of the KDD cycle itself.

4.1 Algorithms

The association rule algorithm used [7], came from Weka [6]. Weka could discretize the continuous values into bins that are then very useful from a generalization standpoint. The a-priori algorithm in the package was used since it provided multiple controls on the confidence of the rules and allowed generation of a selected number of rules. To further develop the attributes chosen, we then used M5 Rules [6], which creates rule sets on continuous data and with the configuration used produces unsmoothed rules with a regression tree built in. Where the association rules look for frequently occurring relationships within attribute ranges, M5 rules dictate that an attribute be considered a class and then looks at the attributes and begins to construct rules that will produce the specific class value.

4.2 Components

The Knowledge Discovery in a Database (KDD) process is typically a nine step process [8]. There is a cyclical nature to those nine steps. This project went through every step to some degree. The data requirements for what is necessary as well as the data acquisition itself have been taken care of already with the data dump from SourceForge.

Because the data dump from SourceForge is in fairly raw form, understanding what data was available, took some time. This by itself was one of the most time consuming steps. The reason for this was that the effort by SourceForge to normalize the database for performance purposes, abstracted away the relationships between the data. That meant certain attributes desired were either not there or broken into multiple tables, requiring more complex SQL queries.

After the redefinition of the problem to pursue activity associations, pre-processing became much easier. We chose to explore the game genre as previously mentioned, which created a smaller data set of 3802 projects. Then the features had to be selected. Using the unique identifier of group ID's allowed us to link multiple attributes including ones that required a SQL statement to aggregate.

The attributes chosen for examination include, number of core developers, number of posts in the forums, how many surveys the groups have used, number of posters to the forums, the number of downloads of the software, and then the rankings both numeric and percentile based for each project. The thought was the SourceForge uses those rankings to identify activity; they are based off of number of times the summary page for a group is viewed and how often the software is downloaded in a given time span. With this as a support, the goal was then to see if these other attributes were associated in the same ways. This represents the data pre-processing steps considered to be part of the KDD process.

The final steps used in the process by this project, were the constructing of the model, and then the analysis. As previously mentioned, using Weka meant that the attributes had to be discretized in order to construct a more nominal structure that the a-priori algorithm could work with, while still using a continuous set of the data for M5 Rules. After experimenting with the few filters that performed binning style discretization on the attributes, we found that every option available did a suitable job of binning the data into bins that would provide a seemingly finite group of now "nominal" values for the association algorithm to work with. The next step was to investigate the minimum confidence of the rules and this played a role in generating more and more rules with lower confidence. Once some associations where built, using the non-discretized set of data, certain attributes could be selected as the examples class, thus the M5 rules would build around trying to predict values for that class.

5. RESULTS

The association results in the raw form are included in Appendix A. The results are set of associations dealing with the binned attributes selected from the database as mentioned above. From viewing the graphs in Appendix C the most noticeable issue observed is that these bins are equal frequency, even with the understanding that one bin will have a disproportionate number of entries, multiple attributes have one bin with a size much larger than the rest of the bins. This is caused by the fact that some stats are missing in the database, indicating that they haven't happened, For example the rankings do not extend forever but at some predetermined lowest activity mark, SourceForge will not rank the project.

Also an attribute like downloads won't be listed in the database for a group if downloads haven't occurred, so when the data was being converted to a form that Weka could read, the number 0 was substituted by a missing value, and for the rankings attribute, an extremely large ranking (100000) was given to projects without one in the database. This however caused some very uneven binning that was too difficult to smooth over.

This problem propagated through the association rules. The majority of the rules were based on already known relationships and all of the rules through a confidence of .6 dealt with the non-active groups. It appeared that the disproportionate amount of inactive groups caused the rules to focus on these groups. In the definition of KDD, one of the main concepts of success is that the results be non-trivial. This obviously failed to qualify.

In order to compensate, we removed any project in which SourceForge did not provide a ranking to the project. The bright side would be that now the only project would be at least moderately active,; however , the reduction in available instances left 756 total, dropping from 3802. This drop produced a smoother discretization for each bin as can be seen in Appendix B. This drop reinforces that fact that while SourceForge has lots of projects worth looking into, for the purposes of exploring community interactions, there is still a great deal of waste.

After running the a-priori algorithm with the new set and its discretization, the algorithm returned with 24 new rules (Appendix A). When analyzed, it is important to note that none of the rules deal with the relationship between the aggregated statistics and the ranking and percentile features provided by SourceForge. In addition to this fact, another observation is that they all have to deal with what seems like a very limited condition of few developers, very few posts and posters, and no surveys. This is probably because out of the remaining instances left, a good proportion of them involve these situations. Also the number of posters and number of posts show up as associating with one another. This once again was a trivial matter and required removing one of the attributes from consideration. After trying to view this problem from a community perspective, it seems that removing posters would be the better choice. A community can still be small and tremendously active if the small group posts a lot. However 10 posters each posting once would be less interesting.

A third round of association rule testing was done with the even smaller sample of data, this time as seen in Appendix D, only 8 rules come to the surface. These 8 rules still focus on the range of small numbers of developers, and posts. Essentially no community activity, but still if SourceForge ranked these projects, something must be happening.

This brings the conversation back to how SourceForge itself prefers to measure activity. As previously mentioned, by searching SourceForge's website, and investigating the activity percentile and ranking metrics, it appears that the percentile is based off of number of times the main page of a group has been visited, presumably on its way to download or post, and the number of downloads. With this knowledge however, we can deduce that with low number of postings and developers, the project will be on the lower end of the activity spectrum and unless the downloads are still high enough to make up for the deficit, the project would not be ranked. Thus, any project that satisfies the conditions listed in the 8 association rules, and has a ranking on SourceForge, most likely is there because of it's downloads. This would mean that to anyone wishing to study the nature of Open Source Group, it would be wise in find the ranked groups, and then by using these rules, eliminate groups without any real community presence, sustained by what may be a finished product, or a very quiet group.

So there are three types of general groups at this stage: totally inactive, those that are active but subsist on downloads, and those which have not only downloads, but what would appear to have communities. To further find how these attributes relate to one another M5 Rules were run using three of the commonly recurring attributes in the association rules as the class. All the original attributes listed will be used in the M5 rule algorithm. The three chosen were number of core developers, number of posters, and number of downloads. In Appendices E-G, the M5 Rules are laid out.

Download was chosen to better understand how it relates to other factors in the community. All the rules it generated use the other two attributes selected plus the number of postings to determine the class value. This would indicate that both core developer size and activity of the forums seem to positively affect the number of downloads.

Core Developers would seem by intuition to be relatively independent of any attribute of the group, and accordingly, M5 rules produced only 4 rules for predicting numbers of core developers. The use of posters and posts makes sense, as the larger a community is at its base (the developers) the more likely the communication will increase. Also the SourceForge percentile attribute is used on a few rules, which may indicate that activity and a larger number of core developers may be correlated. This would serve as subtle affirmation of the 8 association rules. The more active by SourceForge's metric you are, the larger the base community, and that would seem to be what was assumed.

Number of posters produced six rules, and the split occurred with number of posts as the common splitting attribute. While this makes sense and reinforced their correlation and why one of the attributes was removed after the second round of association rules. The one interesting note worth mentioning is that some of the rules have two potential split points for posts. This would seem to indicate that while correlated, the community is no solely based on a larger core developer base and is instead could have outside users.

6. CONCLUSIONS

The effect of this investigation could have some interesting implications for not only further study of the Open Source phenomenon, but for the idea of activity mining in other fields of study. Using the data provided to better understand which examples of the data to look further into, would be very helpful where like with SourceForge, there is a lot of data, and a significant portion of the data is either useless, or can be ignored for certain research purposes. This work could help problems where too much data prevents certain algorithms from being used because of their run time.

We believe that only with sociological experiments and analysis can the true effectiveness of this system be explored and possibly incorporated into a new methodology for software engineering.

Isolating these groups for analysis is important and that is what has begun to become explored by data mining and machine learning fields.

6.1 Future Work

For future work, trying to establish a more formal framework of this model would be a great place to start. Through formalizing the process, one could potentially take this to other sets of data in order to produce smaller more relevant sample. A form of noise reduction that reacts on the right bias for the researcher would be very helpful. Also this is only the tip of the iceberg for understanding this community.

7. ACKNOWLEDGMENTS

The results reported in this paper are based in part upon work supported by the National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829.

REFERENCES

[1] Feller, J., and Fitzgerald, B. Understanding Open Source Software Development Addison-Wesley, London, UK, 2002.

[2] DiBona, C., Ockman, S., and Stone, M. (eds.) Open Sources: Voices for the Open Source Revolution. O'Reilly, Sebastapol, CA, 1999.

[3] Madey, G., Freeh, V., and Tynan, R. "Modeling the F/OSS Community: A Quantitative Investigation," in: Free/Open Source Software Development, S. Koch (ed.), Idea Group Publishing, Hershey, PA, 2004.

[4] J. Xu and G. Madey, "Exploration of the Open Source Software Community", *NAACSOS Conference 2004*, Pittsburgh, PA, June 2004

[5] Y. Gao, Y. Huang and G. Madey, "Data Mining Project History in Open Source Software Communities", *NAACSOS Conference 2004*, Pittsburgh, PA, June 2004

[6] Ian H. Witten and Eibe Frank, "Data Mining: Practical machine learning tools with Java implementations,", Morgan Kaufmann, San Francisco, 2000.

[7] Rakesh Agrawal and Ramakrishnan Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. Very Large Data Bases*, 1994.

[8] Nitesh V. Chawla, "Data Mining Class", http://www.cse.nd.edu/~nchawla

Appendix A:

The 24 rule model:

Best rules found:

1. core_devlopers='(-inf-1.5]' no_posts='(2.5-3.5]' no_survey='(-inf-0.5]' 144 ==> no_posters='(0.5-1.5]' 144 conf:(1)

2. core_devlopers='(-inf-1.5]' no_posts='(2.5-3.5]' 155 ==> no_posters='(0.5-1.5]' 154 conf:(0.99)

3. no_posts='(2.5-3.5]' no_survey='(-inf-0.5]' 274 ==> no_posters='(0.5-1.5]' 271 conf:(0.99)

4. no_posts='(2.5-3.5]' 294 ==> no_posters='(0.5-1.5]' 290 conf:(0.99)

5. core_devlopers='(-inf-1.5]' no_posters='(0.5-1.5]' 189 ==> no_survey='(-inf-0.5]' 177 conf:(0.94)

6. core_devlopers='(-inf-1.5]' no_posts='(2.5-3.5]' no_posters='(0.5-1.5]' 154 ==> no_survey='(-inf-0.5]' 144 conf:(0.94)

7. no_posts='(2.5-3.5]' no_posters='(0.5-1.5]' 290 ==> no_survey='(-inf-0.5]' 271 conf:(0.93)

8. no_posts='(2.5-3.5]' 294 ==> no_survey='(-inf-0.5]' 274 conf:(0.93)

9. core_devlopers='(-inf-1.5]' no_posts='(2.5-3.5]' 155 ==> no_survey='(-inf-0.5]' 144 conf:(0.93)

10. core_devlopers='(-inf-1.5]' no_posts='(2.5-3.5]' 155 ==> no_survey='(-inf-0.5]' no_posters='(0.5-1.5]' 144 conf:(0.93)

11. no_posts='(2.5-3.5]' 294 ==> no_survey='(-inf-0.5]' no_posters='(0.5-1.5]' 271 conf:(0.92)

12. no_posters='(0.5-1.5]' 365 ==> no_survey='(-inf-0.5]' 334 conf:(0.92)

13. downloads='(-inf-0.5]' 94 ==> no_survey='(-inf-0.5]' 85 conf:(0.9)

14. core_devlopers='(-inf-1.5]' 285 ==> no_survey='(-inf-0.5]' 256 conf:(0.9)

15. core_devlopers='(1.5-2.5]' 156 ==> no_survey='(-inf-0.5]' 132 conf:(0.85)

16. core_devlopers='(-inf-1.5]' no_posters='(0.5-1.5]' 189 ==> no_posts='(2.5-3.5]' 154 conf:(0.81)

17. core_devlopers='(-inf-1.5]' no_survey='(-inf-0.5]' no_posters='(0.5-1.5]' 177 ==> no_posts='(2.5-3.5]' 144 conf:(0.81)

18. no_survey='(-inf-0.5]' no_posters='(0.5-1.5]' 334 ==> no_posts='(2.5-3.5]' 271 conf:(0.81)

19. no_posters='(0.5-1.5]' 365 ==> no_posts='(2.5-3.5]' 290 conf:(0.79)

20. core_devlopers='(-inf-1.5]' no_posters='(0.5-1.5]' 189 ==> no_posts='(2.5-3.5]' no_survey='(-inf-0.5]' 144 conf:(0.76)

21. no_posters='(0.5-1.5]' 365 ==> no_posts='(2.5-3.5]' no_survey='(-inf-0.5]' 271 conf:(0.74)

22. core_devlopers='(-inf-1.5]' no_survey='(-inf-0.5]' 256 ==> no_posters='(0.5-1.5]' 177 conf:(0.69)

23. core_devlopers='(-inf-1.5]' 285 ==> no_posters='(0.5-1.5]' 189 conf:(0.66)

24. core_devlopers='(-inf-1.5]' 285 ==> no_survey='(-inf-0.5]' no_posters='(0.5-1.5]' 177 conf:(0.62)

The 8 rule model:

Best rules found:

1. core_devlopers='(-inf-1.5]' no_posters='(0.5-1.5]' 189 ==> no_survey='(-inf-0.5]' 177 conf:(0.94)

2. no_posters='(0.5-1.5]' 365 ==> no_survey='(-inf-0.5]' 334 conf:(0.92)

3. downloads='(-inf-0.5]' 94 ==> no_survey='(-inf-0.5]' 85 conf:(0.9)

4. core_devlopers='(-inf-1.5]' 285 ==> no_survey='(-inf-0.5]' 256 conf:(0.9)

5. core_devlopers='(1.5-2.5]' 156 ==> no_survey='(-inf-0.5]' 132 conf:(0.85)

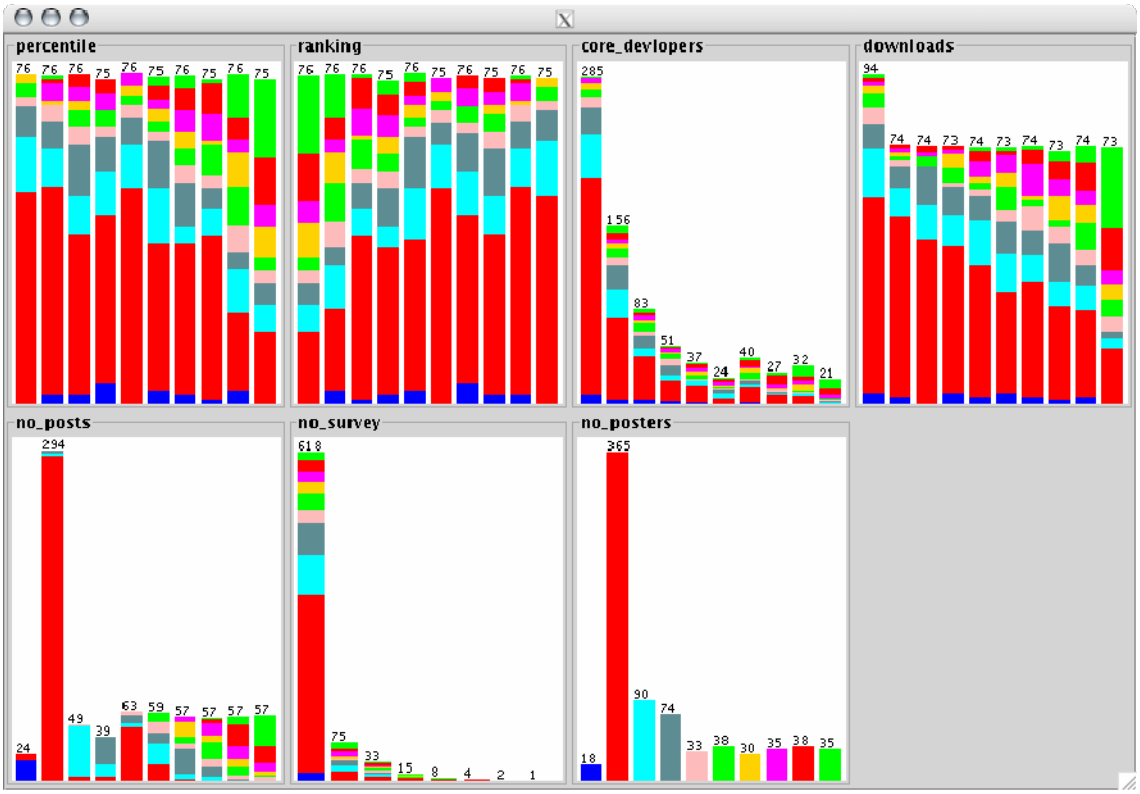
6. core_devlopers='(-inf-1.5]' no_survey='(-inf-0.5]' 256 ==> no_posters='(0.5-1.5]' 177 conf:(0.69)

7. core_devlopers='(-inf-1.5]' 285 ==> no_posters='(0.5-1.5]' 189 conf:(0.66)

8. core_devlopers='(-inf-1.5]' 285 ==> no_survey='(-inf-0.5]' no_posters='(0.5-1.5]' 177 conf:(0.62)

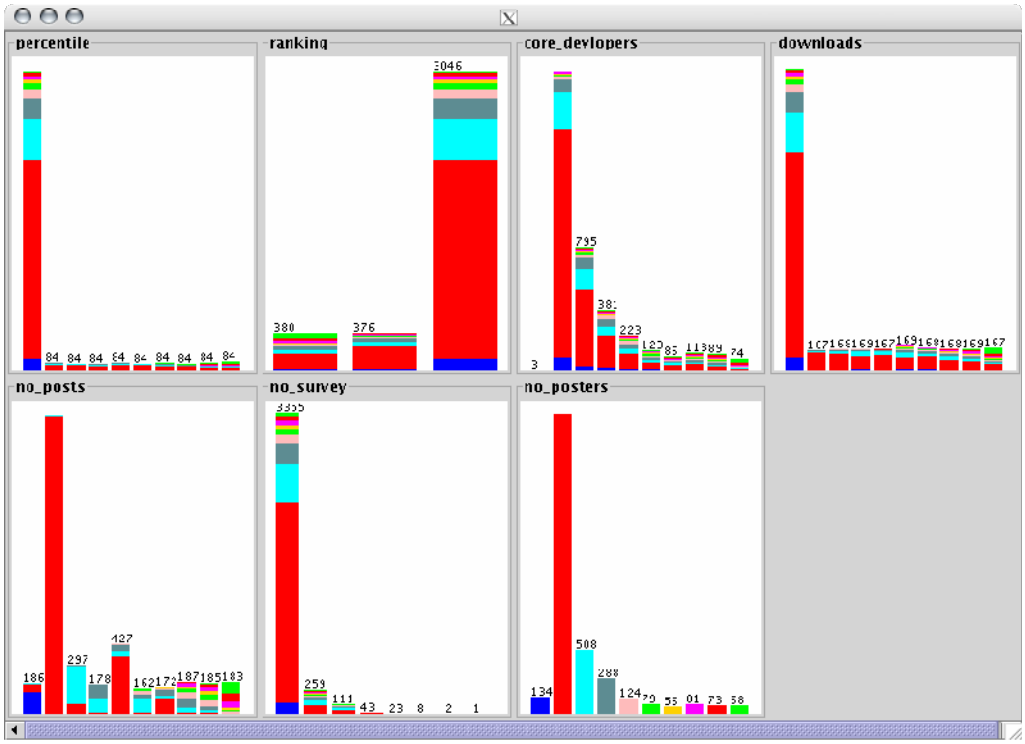
Appendix B:

The Binned graphs for the final model



Appendix C:

The Binned attributes before reduction of non-actives



Appendix E:

The M5 Rules for downloads:

M5 pruned regression rules :
Number of Rules : 5

Rule: 1
IF
	no_posters <= 5.5
	core_devlopers <= 2.5
THEN

downloads =
	+ 2611.1884 [398/45.996%]

Rule: 2
IF
	no_posters <= 11.5
	core_devlopers <= 6.5
	no_posts <= 29.5
THEN

downloads =

+ 4212.8614 [166/29.405%]

Rule: 3
IF
no_posters <= 22.5
THEN

downloads =
+ 15037.6755 [151/62.846%]

Rule: 4
IF
core_devlopers <= 16.5
THEN

downloads =
+ 54154.8125 [32/62.09%]

Rule: 5

downloads =
+ 331176.4444 [9/105.433%]

Appendix F:

M5 Rules on Core Developers

M5 pruned regression rules :

Number of Rules : 4

Rule: 1

IF

no_posters <= 5.5

percentile <= 47.086

THEN

core_devlopers =

+ 2.1167 [300/62.713%]

Rule: 2

IF

no_posters <= 9.5

no_posts <= 7.5

percentile <= 82.764

THEN

core_devlopers =
	+ 2.4972 [181/54.361%]

Rule: 3
IF
	no_posters <= 11.5
	percentile <= 91.618
	no_posts <= 45.5
THEN

core_devlopers =
	+ 3.7967 [123/61.244%]

Rule: 4

core_devlopers =
	+ 9.2434 [152/153.894%]

Appendix G:

M5 Rules for Posters:

M5 pruned regression rules :
Number of Rules : 6

Rule: 1
IF
	no_posts <= 21.5
THEN

no_posters =
	+ 1.8462 [598/14.326%]

Rule: 2
IF
	no_posts <= 232

downloads <= 5967
no_posts <= 102.5
THEN

no_posters =
+ 7.0667 [60/7.718%]

Rule: 3
IF
no_posts <= 766
no_posts <= 232
THEN

no_posters =
+ 15.4333 [60/16.409%]

Rule: 4
IF
no_posts <= 912
THEN

no_posters =
+ 34.25 [20/33.798%]

Rule: 5
IF
no_posts <= 4075.5
THEN

no_posters =
+ 143.2143 [14/65.572%]

Rule: 6

no_posters =
+ 416 [4/103.622%]

Appendix H:

The Statistical distribution of the 756 example set

percentile

Min	Max	Mean	StdDev
.089	99.872	52.658	29.415

ranking

Min	Max	Mean	StdDev
14	10151	4810.467	2988.294

core developers

Min	Max	Mean	StdDev
1	67	3.914	5.624

# of downloads

Min	Max	Mean	StdDev
0	602501	11538.12	51430.837

# of posts

Min	Max	Mean	StdDev
0	8889	102.869	593.901

# of surveys

Min	Max	Mean	StdDev
0	10	.354	.995

# of posters

Min	Max	Mean	StdDev
0	623	9.005	40.351