

UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Software Engineering Curriculum

Myroslava Stavnycha
Issue Report Resolution Time Prediction
Master's Thesis (30 ECTS)

Supervisor(s): Dietmar Pfahl

Tartu 8/4/15

Abstract:

Prediction of the resolution time of an issue report has always been an important, but difficult, task. The primary purpose of this study is to build a model that predicts the resolution time of incoming issue reports based on past issue report data. Moreover, additional goals of the research are to determine which existing approaches of resolution time prediction yield the highest levels of accuracy, and which features of issue reports are essential for prediction. The approach chosen for building an issue resolution time prediction model was to improve currently existing models applying additional reports pre-processing. The project was designed to analyse, combine, compare and improve different techniques of resolution time prediction. This includes k-means clustering, k-nearest neighbor classification, Naïve Bayes classification, decision trees, random forest and others, in order to achieve the best results with regards to prediction accuracy. For conducting the current research, data was collected from a repository of the Estonian company Fortumo OÜ. The data provided by Fortumo contained actual resolution times of 2125 issues from 25 Apr 2011 till 1 Jan 2015 along with initial time estimates made by Fortumo employees.

The data from the repository indicates that around 50% of the time estimates made by Fortumo employees fall into the range of $\pm 10\%$ of the actual resolution time. In addition, 67% of experts' estimates have absolute error ≤ 0.5 hour. Existing proposed approaches don't increase the predictive quality. On the contrary, proposed methods bring worse results. Random Forest and Ordered Logistic Regression, as the best among the proposed models, still produced a prediction quality 12-20% worse than the estimates of the experts. After improvement of the best performing approaches, meta-information-based models yielded a better accuracy than proposed models by up to 5%. However, text-based models produced a higher prediction quality, approximately up to 20% better than estimates made by experts.

Keywords:

Machine learning, data mining, prediction, k-means, k-nearest neighbours, random forest, ordered logistic regression, Naïve Bayes classifier, latent semantic analysis, issue report, resolution time

Lühikokkuvõte:

Ennustus ajakulu kohta probleemi teatamise ja lahendamise juures on alati olnud tähtis kui samas raske ülesanne. Peamine eesmärk selle töö juures on ehitada modell mis ennustab eelnevate aruannete andmete põhjal probleemi lahendamiseks ja tulemuste saamiseks kuluvat aega. Lisaks täiendavad eesmärgid uurimuse juures määravad millised meetodid on kõige kõrgema usaldusväärsusega ning millised funktsioonid on olulised ennustuseks. Eesmärk miks valiti probleemi lahendamise ajakulu modell oli edasi anrendada juba olemas olevaid modelle lisades erinevaid lisasid. Projekt loodi analüüsimaks, kombineerimaks, võrdlemaks ja edendamaks erinevaid tehnikaid probleemi lahendamise ennustamisel. See sisaldab k-means klastreid, k-nearest neighbor klassifikatsiooni, Naïve Bayes klassifikatsiooni, otsustus puid, juhuslikku metsa ja teisi, parima tulemuse saamiseks. uurimuse läbiviimiseks koguti andmed Eesti firmalt Fortumo OÜ. Fortumo andmed sisaldasid 2125 probleemi lahendamise aegasid alates 25 aprillist 2011 aastal kuni esimese jaanuarini 2015 aastal. koos kommentaaridega Fortumo töötajatelt.

Andmed näitasid et 50% ajakuludest mis Fortumo töötajad märkisid olid vahemikus $\pm 10\%$ tegelikust ajakulust. Lisaks 67 % nendest omavad kindlat viga ≤ 0.5 tunni võrra. Olemasolevad ettepanekud ei tõstnud probleemi lahendamise kvaliteeti. Vastupidiselt töid hoopis halvemaid tulemusi. Juhuslik mets ja tellitud logistiline regressioon olles parimad nimetatute hulgas näitasid siiski kuni 12-20% halvemat tulemust kui ekspertide omad. Pärast parimate võimaluste täiendamist, meta-informatsiooni modellid näitasid paremat sobivust kuni 5% võrra. Kuigi, tekstil põhinevad medellid andsid kõrgema kvaliteeti, umbes 20% kõrgema kui ekspertidel.

Märksõnad:

Masina õpe, data mining, ennustus, k-means, k-nearest neighbours, juhuslik mets, tellitud logistiline regressioon, Naïve Bayes klassifikatsioon, varjatud semantiline analüüs, probleemi reporteerimine, lahendamise aeg.

Table of Contents

List of Abbreviations	6
1 Introduction.....	8
1.1 Problem Statement	9
1.2 Structure.....	10
2 Current Practice of RT Prediction in Fortumo.....	11
2.1 The Process of Estimating Issues in Fortumo.....	11
2.2 Calculating Prediction Quality in Fortumo.....	11
3 Related Work	15
3.1 Prediction Models	15
3.2 Feature Selection for Prediction Model	21
3.3 Removing Outliers for Improving Model Accuracy.....	23
4 Application of Recommended Models to Fortumo Data.....	27
4.1 Issue Report Description.....	27
Issue Reports Extraction and Selection.....	27
Issue Report Attributes Description.....	28
4.2 K-Nearest Neighbors	29
4.3 Naïve Bayes Classifier.....	32
4.4 C4.5 Decision Tree	32
4.5 Random Forest	33
4.6 Ordered Logistic Regression.....	33
4.7 Other Methods	34
4.8 Summary.....	34
5 Case Study Elements.....	36
5.1 Moving Window Concept.....	36

5.2	Meta-Information-Based Model	37
	Feature Selection.....	38
	Removing Mild Outliers	38
5.3	Text Based Model	38
	Preprocessing Textual Data	39
	Calculating the Distance Between Documents	40
	Latent Semantic Analysis	40
	Removing Mild Outliers	41
	Improved Spherical K-means Clustering.....	41
6	Case Study Execution and Results.....	44
6.1	Enhancement of Accuracy of Meta-Information Based Model Prediction	44
	Feature Selection.....	44
	Model Application Results	49
6.2	Enhancement of Accuracy of Text-Based Model Prediction	52
6.3	Discussion	58
7	Conclusion	63
8	Bibliography	64

List of Abbreviations

AP	Actual Prediction Accuracy
AE	Absolute Error
AUC	Area Under the Curve
BP	Baseline Prediction Accuracy
CDT	C/C++ development tools
COCOMO	Constructive Cost Model
df	Degrees of Freedom
GEF	Graphical Editing Framework
IDF	Inverse Document Frequency
IEEE	Institute of Electrical and Electronics Engineers
IQ	Inter-Quartile
JDT	Java Development Tools
kNN	K Nearest Neighbors
LSA	Latent Semantic Analysis
MMRE	Mean Magnitude of Relative Error
MRE	Mean of Relative Error
NASA	National Aeronautics and Space Administration
ORL	Ordinal Logistic Regression
OS	Operating System
PCA	Principle Component Analysis
PDE	Plug-in Development Environment
PM	Project Manager
PRED	Predictive Quality
Q	Quartile

RE	Relative Error
RF	Random Forest
RT	Resolution Time
SLIM	Software Lifecycle Management
SLOC	Source Lines of Code
SOM	Self-Organizing Maps
SVD	Singular Value Decomposition
TF	Term Frequency
WRO	Without Removing Outliers
α -kNN	α -K Nearest Neighbors

1 Introduction

Nowadays, planning and scheduling is critical for companies of any size. We use planning in order to know how much a product will cost, how much resources are needed, and when a product will be delivered. Estimating and planning is an integral part of the software development process. It is important for the overall success of a project, as it determines the feasibility of said project. Business decisions, tactics, and actions like scheduling marketing campaigns, demo presentations, releases, and advertisements rely on dates and deadlines that are predicted. Plans help us know if a project is on track to deliver the functionality that user expects. Thus, planning reduces risks and uncertainty.

The process of planning helps developers to better understand what should be built and which tools to apply in order to achieve a higher performance. It is a process of searching for an optimal solution between features and resources. Planning and estimates are used to support decision-making. They help to understand whether a project should or should not be implemented. However, planning is difficult and plans are often wrong. Teams often tend to respond to this by either not doing planning at all or by putting so much efforts into planning that there is no time left for actual work. Often, estimations are not valid or well-grounded. Moreover, people can often be influenced by other people's opinion or other subjective factors that can skew the estimation. In addition, developers tend to assign an optimistic estimate to a feature. An optimistic estimate does not cover unexpected circumstances, additional communication with colleagues, problems with tools, etc. Also, the amount of time necessary to execute tests or some previous code improvement is often overlooked or not taken into account, even though it is still required. Given a recurring task in a project, people often tend to forget to check historical data in order to improve time prediction, but instead assign another guess estimate for the resolution of that task even though a guess estimate has been previously ascertained for a similar task. According to the data used in this thesis, which was provided by Fortumo, only up to 20% of time estimates fall into the range of $\pm 10\%$ of an actual resolution time. Thus, it is clearly visible that the accuracy can be improved.

Some models, such as COCOMO, SLIM and CheckPoint were developed in order to define resolution time of a feature, resources and cost. They are mostly based on function point analysis and integral features of a team. However, for highly accurate prediction one must provide accurate input, which complicates the task.

However, development of a product is expensive and as a result project stakeholders put a lot of pressure on both the project manager (PM) and developers' team. This pressure affects the project quality. Another issue that affects project quality is wrong or faulty estimation. Developers follow these estimates and when they infer their inability to deliver before the final deadline, begin to cut corners thus reducing the quality.

In order to reduce the discrepancy between the predicted and actual time, companies tend to move to estimation of size i.e. story points. However, for business, this metric is not as simple as time estimation.

Today, people possess huge amount of data, which they do not analyze or use for any purpose. Previously several researchers have made contributions towards transforming existing data into a decision-making support for predicting resolution time. These studies however, did not bring robust enough results. The ones with acceptable accuracy rates often suffer from optimistic bias and overfitting, and were eventually disproved by other researchers. In addition, they employed different measures of calculating prediction accuracy and conducted their studies on different data. Consequently, the results are not comparable. As a result, it is hard to find the best recommended prediction model.

1.1 Problem Statement

The objective of this thesis is to compare existing studies using our own measure of prediction accuracy. In addition, we aim to improve existing approaches and combine best practices in order to outperform existing models and build a much more reliable model to streamline the development process for all engineering teams, namely simplifying the planning process and guaranteeing reliable estimation.

Furthermore, I will explore which level of accuracy can be derived from existing data.

For measuring prediction accuracy, different quality measures will be used, based on both absolute and relative error.

Thus, the main set of research questions in this thesis is:

1. What is the current RT prediction accuracy at Fortumo?
2. What is the accuracy of proposed (existing) RT prediction models applied to Fortumo data?
3. How can the best performing existing RT prediction models be improved?

4. What is the prediction accuracy of improved RT prediction models applied to Fortumo data?

In addition, in this thesis we divide models of resolution time prediction into two categories:

1. Meta-information-based model (Type 1).
2. Text-based model (Type 2).

1.2 Structure

The current thesis is structured as follows:

Section 2 reviews the current process of resolution time prediction in Fortumo. In addition, it defines measures for defining prediction accuracy and calculates the prediction accuracy of experts' estimates in Fortumo.

Section 3 presents a set of recommended techniques for estimating issue report handling proposed in the literature.

Section 4 applies all recommended models to Fortumo data in order to compare their accuracy using defined metrics. Moreover, the model, which gives the highest accuracy, is set as the *baseline model* with which all improved approaches for predicting the resolution time proposed in this thesis will be compared.

Section 5 presents a plan of researching additional approaches for estimation of the resolution time of an issue report.

Section 6 describes the process of applying proposed techniques of resolution time prediction on Fortumo data. It also presents the results using defined metrics for measuring accuracy of the prediction. Moreover, an additional discussion about the results and future work is presented in this section.

Section 7 concludes the thesis.

2 Current Practice of RT Prediction in Fortumo

In order to understand the current situation of prediction accuracy in Fortumo, we studied the process of RT prediction at Fortumo and measured its accuracy. As a result, 67% of estimations were correct within ± 0.5 hour of the actual resolution time. In addition, half of predictions were correct within $\pm 10\%$ of the actual resolution time.

2.1 The Process of Estimating Issues in Fortumo

Time estimation of incoming issues is done on a weekly basis during meetings. It involves the opinion of the whole team of developers, who are in charge of the issue. Usually the procedure follows Planning Poker rules, which is an agile software development practice [1].

2.2 Calculating Prediction Quality in Fortumo

In order to examine the actual situation and evaluate the accuracy of prediction done in Fortumo by its employees, we analyzed existing data and calculated its predictive quality. Issue reports extracted from Fortumo's repository, contained such attributes as *resolution_time* and *time_estimate*, measured in seconds which corresponded to time spent on the issue and initial estimated time which represent theoretical time that issue should take.

We convert initial time prediction of the issue report to hours and then to our discrete scale, which is defined in the following way:

1. [0; 0.5]
2. (0.5; 1]
3. (1; 3]
4. (3; 6]
5. (6; 11]
6. (11; 20]
7. (20; 40]
8. (40; $+\infty$)

In this study, we assume that a given set of classes of RT gives enough information about RT for practical work. The distribution of resulting RT classes is described in Figure 2.1.

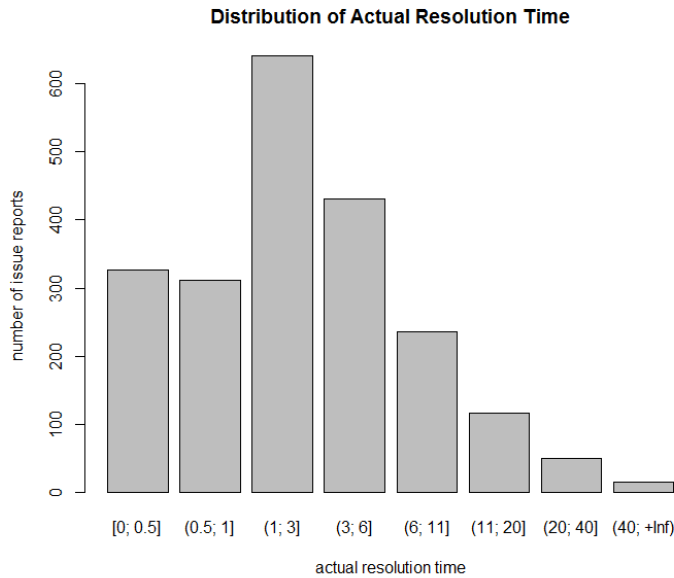


Figure 2.1 Distribution of actual resolution time in Fortumo

Out of 2125 issues, there are 894 issues with RT estimates.

For issues with an RT estimate, the distribution of their estimate and actual RT is shown in Figure 2.2.

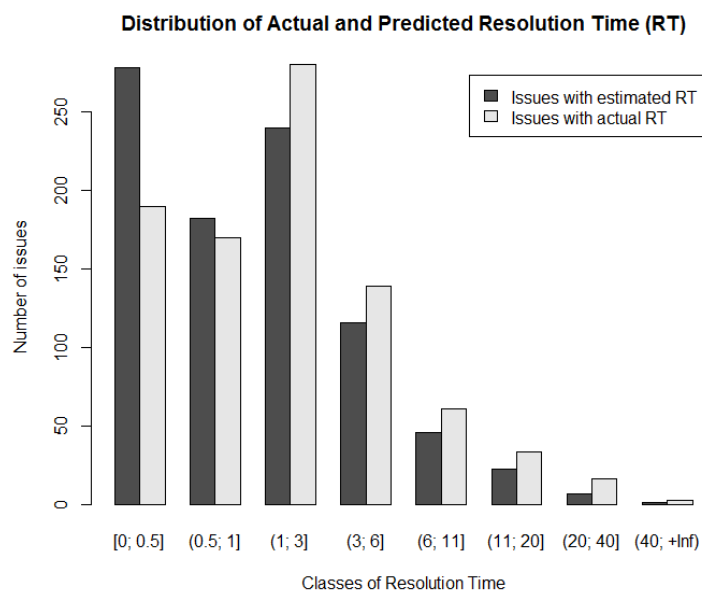


Figure 2.2 Distribution of estimated RT and actual RT

Figure 2.2 depicts that experts tend to underestimate the resolution time of the issues since the distribution of estimated RT is more skewed to the left than the distribution of actual RT.

In our study, we used 2 kinds of prediction accuracy measures:

1. Predictive Quality using Absolute Error (AE).
2. Predictive Quality using Relative Error (RE).

Absolute Error. Absolute error is defined as absolute difference between predicted value and actual value:

$$AE = \text{difference}(\text{predicted}, \text{actual})$$

Seeing that the predicted resolution time is an interval value, the difference between actual RT and the predicted interval of RT is defined as follows:

$$\text{difference}(rt_p, rt_a) = \begin{cases} 0, & \text{if } rt_a \in [rt_{p_{min}}; rt_{p_{max}}] \\ \min(|rt_{p_{min}} - rt_a|, |rt_{p_{max}} - rt_a|), & \text{otherwise} \end{cases}$$

Where rt_p is the predicted interval of resolution time, rt_a is the actual resolution time in hours.

Relative Error. Relative Error is defined as the division of Absolute Error by the actual RT:

$$RE = \frac{AE}{rt_a}$$

Predictive Quality using Absolute Error. This approach of evaluating predictive quality is calculated as a percentage of issues with Absolute Error $\leq X$:

$$Pred(X) = \frac{\sum_i (AE(d_i) \leq X)}{|D|}$$

Where X is an Absolute Error, D is the documents set, $d_i \in D$.

Predictive Quality using Relative Error. This way of estimating quality of prediction is defined as a percentage of issues with Relative Error $\leq X$:

$$Pred(X) = \frac{\sum_i (RE(d_i) \leq X)}{|D|}$$

Where X is Relative Error, D is the document set, $d_i \in D$.

In this study, we consider only Pred(0.5h) and Pred(1h) as measures for assessing the model in terms of absolute error since the mean RT of issue reports in Fortumo’s dataset is 4.8 hours and median class of RT is (1; 3]. Thus, these metrics are sharp enough and depict substantial information about model accuracy. Predictive Quality from Absolute Error is especially valuable for issues with large resolution time. However, since we still have issues with large resolution times, we use the Predictive Quality from Relative Error; namely Pred(10%) and Pred(25%), as we consider those measures to be strict enough as well.

Using preceding formulas, the calculated Predictive Quality of estimations in Fortumo is described in Table 2.1.

Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)
0.668	0.727	0.501	0.578

Table 2.1 Predictive quality of time estimates in Fortumo

3 Related Work

In order to investigate existing models and their prediction accuracy, we accessed IEEE Xplore Digital Library and ACM Digital Library, as trusted sources of high quality studies. The development of RT prediction system for issue report is not a trivial problem and has been studied for over 40 years. One of the earliest and most popular systems in this area is regression-based COCOMO (CONstructive COst MOdel) [2], which is used for project effort and time estimation, but not single issue reports estimation. COCOMO accumulates a broad set of different project parameters. Its newer version COCOMO II has parameters divided into categories: Software Scale Drivers, Software Cost Drivers Product, Personnel, Platform, Project and Sizing Method (function points or SLOC) [3]. Researchers have tried for many years to improve the prediction accuracy of COCOMO [4] [5] [6] [7] [8]. Unfortunately, C. F. Kemerer in his study [9], showed that COCOMO failed to reflect the dependence of project duration and effort consumption on the considered factors.

This section is divided into 3 parts:

1. Prediction models;
2. Feature selection for prediction models;
3. Removing issue report outliers for improving model accuracy.

3.1 Prediction Models

Thomas Zimmermann, researcher at Microsoft Research, has conducted a study in [10] on JBoss dataset in order to predict fixing effort using the k-Nearest Neighbour Approach, because it is easy and flexible in use. He performed his research on a set of bug reports, which is a subset of issue reports. In order to construct a similarity measure, two attributes of bug reports, *description* and *title*, were selected. Since text similarity measure is crucial for current research, the authors of the paper used a text similarity measuring engine known as Lucene, a product of Apache. The authors' results indicated poor predictive quality using the kNN approach and the description statistic of results is shown on Figure 3.1, where the difference between actual and predicted time on average is 20h and only 30% lie within $\pm 50\%$ range of the actual effort.

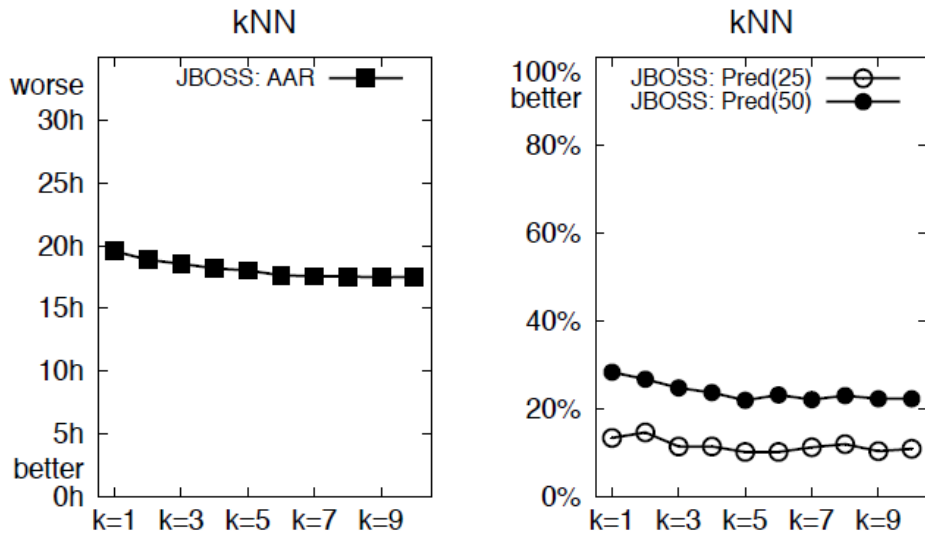


Figure 3.1 kNN performance [10]

Another approach used by Zimmermann in [10] was the α -kNN method with $k = \infty$ and α from 0 to 1 with 0.1 step. Using this approach, the authors theorised that the lower the rate of α is, the better accuracy we obtain (even up to 100% accuracy), which is shown in Figure 3.2.

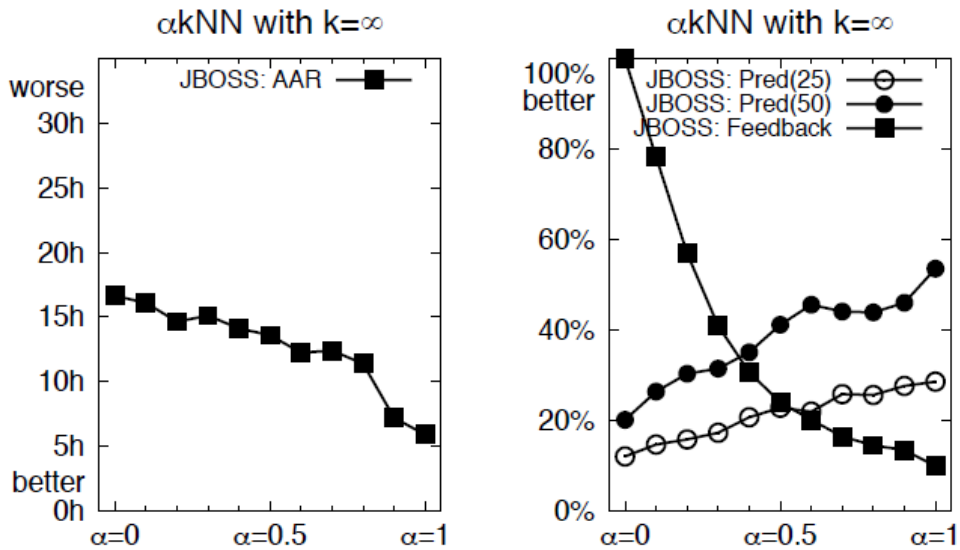


Figure 3.2 α -kNN performance [10]

In [11], Uzma Raja suggested using clustering in order to analyze usefulness of textual data of a bug report for predicting the RT. SAS Text Miner [12] was used for text preprocessing and

clustering. The results of a study showed the statistically significant difference in the means and medians of the RTs between different clusters. As a result, Raja concludes that text-based clustering can be useful for prediction the resolution time.

In [13], researchers from University of Zurich achieved high rates of accuracy of RT prediction using decision tree model when categorizing issues into two groups: ‘Fast’ and ‘Slow’, which stand along two sides of the distribution median. They used data from Eclipse, Mozilla and Gnome projects and built their model using decision tree covering two cases, i.e., initial data (*reporter, date, nextRelease, hToLatFix*) and post-submission data (*assignee, platform, OS, priority, severity, status, comments, milestone* and others), where post-submission data is set of entries of issue report which can be changed after the issue report has been submitted. The results indicated higher accuracy after inclusion of post submission information. Common significant predicators among all projects in case of considering post-submission data appeared to be *milestone, priority, assignee* and *reporter*. While considering only initial data, *date* and *assignee* had the most significant influence.

Project	Precision
Eclipse JDK	0.635
Eclipse Platform	0.654
Mozilla Core	0.639
Mozilla Firefox	0.608
Gnome GStreamer	0.646
Gnome Evolution	0.628

Table 3.1 Decision tree model accuracy for model with initial data [13]

According to Table 3.1, 60-70% of incoming bug reports were correctly predicted. It improves random classification by 10-20%.

Researchers from the George Mason University in [14] applied unsupervised learning of self-organizing maps on NASA IV&V Facility Metrics Data Program repository data. The input to the SOM algorithm was a dissimilarity matrix based on a set of issue attributes such as *severity, how_found, mode, problem_type*. Using Mean Magnitude of Relative Error as a measurement

of accuracy, they obtained results with average MRE in the range of 7% - 23% of RT. The maximum MRE is in the range of 23% - 83% of the actual RT. However, the dataset that covers completely different development environments, was less suitable for the given model and returned an average MRE in the range of 40% - 159% of RT, a maximum MRE ranging from 159% to 373%, which indicates poor model performance.

Lucas D. Panjer from University of Victoria based his study [15] on Eclipse BugZilla data, where he compared five different modelling approaches: 0-R, 1-R, C4.5 Decision Tree, Naïve Bayes Classifier and Logistic Regression.

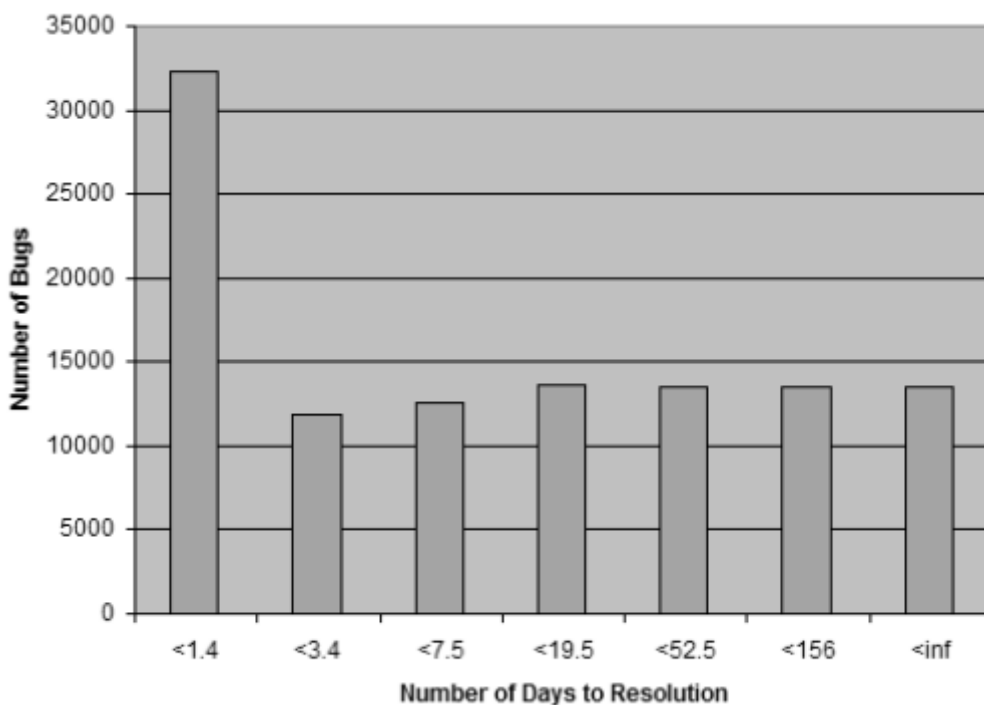


Figure 3.3 Distribution of actual resolution times [15]

In Figure 3.3, the distribution of issue RT is depicted. Reports were divided into 7 clusters according to their bug RT using equal-frequency binning algorithm. The biggest cluster contains reports with bug RT less than 1.4 days.

First, 0-R and 1-R approaches were applied as the definition of baseline classification. While the 0-R approach takes the mode of the distribution as predicted value, 1-R generates 1-level decision tree for every attribute picking up the majority output class for every branch. It then chooses the tree with minimum error to return predicted value. As a result, 0-R returned 29.1%

of correctly classified reports with kappa statistic 0. 0-R predicted all reports with < 1.4 resolution time correctly, since a value < 1.4 is the most likely outcome. 1-R correctly classified 31.0% of data with kappa statistic 0.0747. 1-R algorithm built a 1-level decision tree with comments as its determinant attribute (Figure 3.4)

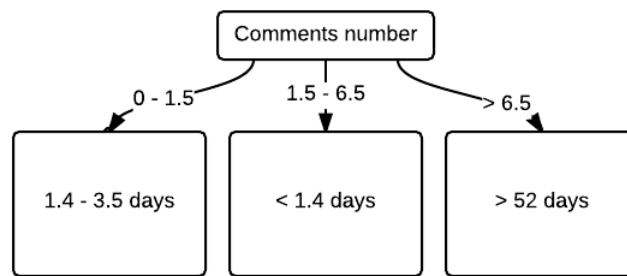


Figure 3.4 Decision tree [15]

The C4.5 decision tree algorithm correctly categorized 31.9% of issues with kappa statistic 0.0938. The top node of the C4.5 tree is always comments with followed assignee attribute.

The Naïve Bayes algorithm produces a higher result with 32.5% of correctly categorized data with kappa statistic 0.1195.

Due to computational constraints for logistic regression, only 469 (0.42% of original dataset) issues were taken into account. However, the given approach correctly defines bug RT for 34.9% of bugs and the kappa statistic is 0.1577.

Algorithm	Predicted %	Kappa
0-R	29.1%	0.0000
1-R	31.0%	0.0747
C4.5 Decision Tree	31.9%	0.0938
Naive Bayes	32.5%	0.1195
Logistic Regression	34.9%	0.1577

Table 3.2 Algorithms results [15]

Logistic Regression applied on the same set of data, produced the best results reaching 34.9% of issues correctly predicted.

Naïve-Bayes classifier was also studied in [16], using data of Eclipse JDT, Mozilla and Gnome projects. However the output resolution time set was divided into two categories:

1. *Fast* and *Slow*, divided by a specified median.
2. *Very Fast* and *Not Very Fast*, divided by 1st quartile.
3. *Not Very Slow* and *Very Slow*, divided by 3rd quartile.

The input to the algorithm is a set of all issue attributes, as: *date*, *severity*, *reporter*, *platform*, *OS* and so on.

Target	Eclipse JDT		Mozilla		Gnome	
	Precision	Recall	Precision	Recall	Precision	Recall
<i>Very fast</i>	0.39	0.20	0.43	0.20	0.76	0.99
<i>Not very fast</i>	0.77	0.90	0.77	0.91	1.00	0.89
<i>Fast</i>	0.57	0.64	0.61	0.65	0.62	0.67
<i>Slow</i>	0.58	0.51	0.62	0.58	0.64	0.59
<i>Not very slow</i>	0.78	0.93	0.81	0.85	0.79	0.85
<i>Very slow</i>	0.49	0.21	0.47	0.41	0.41	0.23

Table 3.3 Results of Naïve-Bayes classifier [16]

Table 3.3 shows that for classes divided by the specified median, the precision of a prediction varies between 57% - 64%. However, when the output set is divided by 1st or 3rd quartiles and the output distribution becomes more skewed, the precision of the prediction accuracy of a target subset with smaller volume becomes worse.

Random forest, as another supervised classifier, was applied to Mozilla and Eclipse datasets in [17] by researchers from Queen's University of Canada. Having the output resolution time set divided into categories < 3 months, < 1 year and <3 years, the authors showed that the current model can produce approximately 65% of correct issue classification. In addition, the creation date and location of an issue has a strong impact on resolution time in contrast to issue priority

which doesn't have any significant influence. As mentioned earlier, in [13], researchers proved the correlation between 'open date' and RT of an issues.

3.2 Feature Selection for Prediction Model

In [18], researchers from Microsoft Research and Stanford University conducted a research which revealed that reports reported by people with higher reputation are more likely to be fixed earlier. The authors used datasets of Windows Vista and Windows 7 in their research and found out the linear dependency between a bug's RT and reporter's reputation. The definition of a reporter's reputation was derived from the number of completed tasks, reported by this person. In Figure 3.5, a clear, consistent, and monotonic increase in bug resolution likelihood as the opener reputation increases, is described.

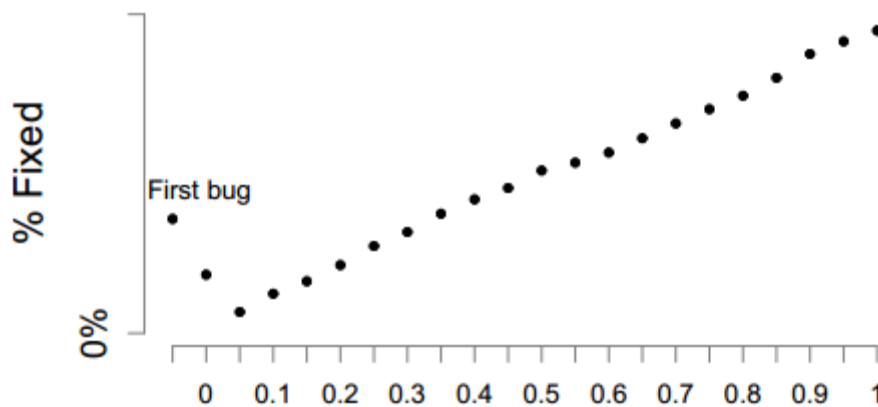


Figure 3.5 Percent of fixed Vista bugs vs. bug reporter's reputation [18]

However, the concept above was disproved in [19], where the same method was applied to another dataset and results revealed no correlation between the values contrasted in the table above.

Additionally, in [19], the authors investigated which attributes of bug reports predict the resolution time better using multivariate regression testing where the dependent variable is bug RT and the independent variables are attributes of a report: *bug severity*, *number of attachments*, and *number of developers* involved. They conducted the research using datasets of Chrome, Mozilla and Eclipse. As a result, a low prediction quality of the model was received, where multivariate goodness of fit, R^2 , was in a range of 30% - 49%, which means that there

is a need for more independent variables in order to construct a better prediction model to predict bug report resolution time.

Project	Adjusted R^2	F-Value	p-value	p-value for independent variables			
				Number of developers	Severity	Attachments	Dependencies
Firefox	0.401	1857.51	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
Thunderbird	0.498	985.11	<0.0001	<0.0001	<0.0001	<0.0001	0.0292
Seamonkey	0.366	2473.95	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
Eclipse	0.301	7079.16	<0.0001	<0.0001	<0.0001	<0.001	<0.0001

Figure 3.6 Multivariate regression testing results [19]

In [20], researchers proved a strong linear correlation between the number of participants and resolution time, based on data pulled from 9 releases of Ubuntu, which is described in Figure 3.7.

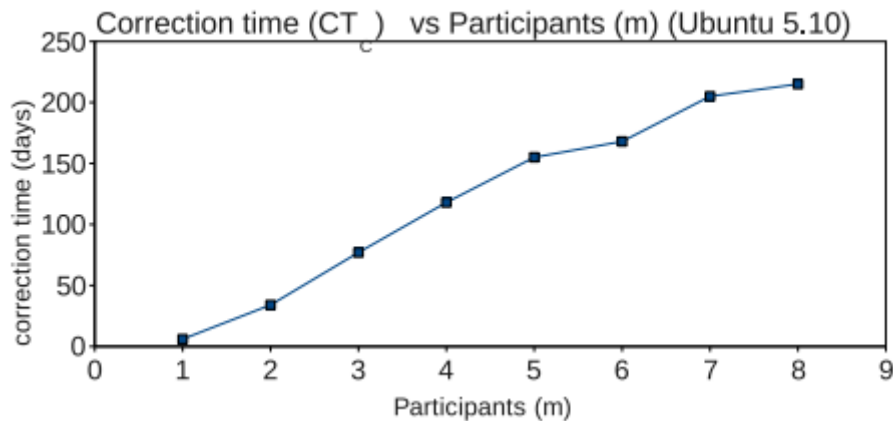


Figure 3.7 Resolution time with respect to participants [20]

The calculated average correlation coefficient is 0.92, which indicates a strong dependence of resolution time on the number of participants. The authors stated that the model, based on this theory, produces high accuracy results:

MMRE	PRED(0.25)
------	------------

0.1 – 0.22	0.7 – 0.8
------------	-----------

Table 3.4 Results of the model, based on number of participants

Where MMRE is Mean Magnitude of Relative Error among all Ubuntu Releases, and PRED(0.25) is the percentage of issues with $MMRE \leq 0.25$. However, the model built in [10] which was based on a kNN approach, produces the same accuracy results and the other, based on an α -kNN approach, slightly outperforms the former as it involves only initial data of the issue report. Additionally, in [19], no significant correlation between the number of participants and resolution time could be found.

In conclusion, all materials presented in this section serve as additional data about the influence of different independent variables on the issue report RT. In this thesis such dependency is analyzed for further feature selection for improving the accuracy of the prediction models.

3.3 Removing Outliers for Improving Model Accuracy

Ahmed Lamkanfi and Serge Demeyer from the University of Antwerp in their paper [21] emphasize the fact that open source RT data is heavily skewed and includes non-realistic data with RT less than a minute. Thus, such outliers may confuse data mining techniques and produce distorted results. Consequently, the authors claim that removing outliers will have positive impact and improve classifiers. The authors used data examined in [13] and compared the results of [14] with the ones after removing outliers.

Project	Minimum	Median	Maximum
Eclipse Platform	10 seconds	8.5 days	9.1 years
Eclipse PDE	12 seconds	4.7 days	5.9 years
Eclipse JDT	10 seconds	4.3 days	7.9 years
Eclipse CDT	9 seconds	8.8 days	7.2 years
Eclipse GEF	8 seconds	13 days	7.2 years
Mozilla Core	11 seconds	13.8 days	11.5 years
Mozilla Bugzilla	3 seconds	2.7 days	10.7 years
Mozilla Firefox	13 seconds	7.6 days	11.1 years
Mozilla Thunderbird	18 seconds	32 days	10.3 years
Mozilla Seamonkey	14 seconds	5.8 days	12.7 years

Table 3.5 Descriptive statistic of datasets resolution time [21]

As Mozilla developers explained in [21], issue report RT can take more than 100 days in cases of insufficient information, incorrect description or specifying wrong component of the software system. Unfiltered RT distributions of Eclipse and Mozilla data are presented in Figure 3.8 and Figure 3.9.

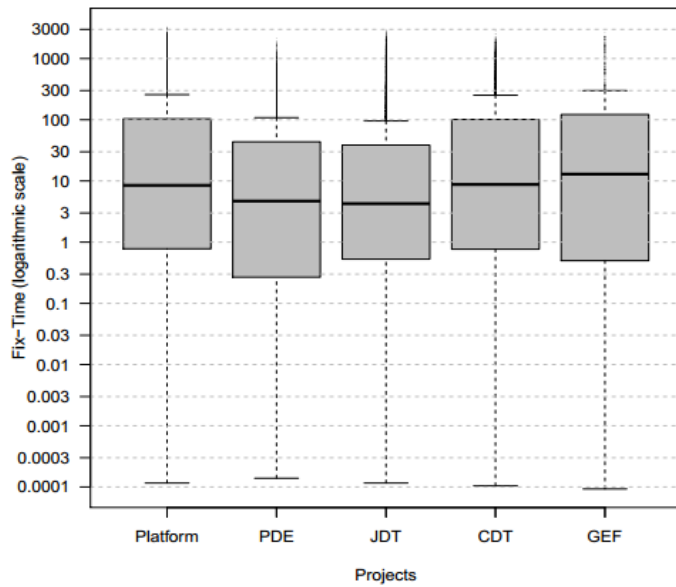


Figure 3.8 Boxplots of RT in days of Eclipse projects [21]

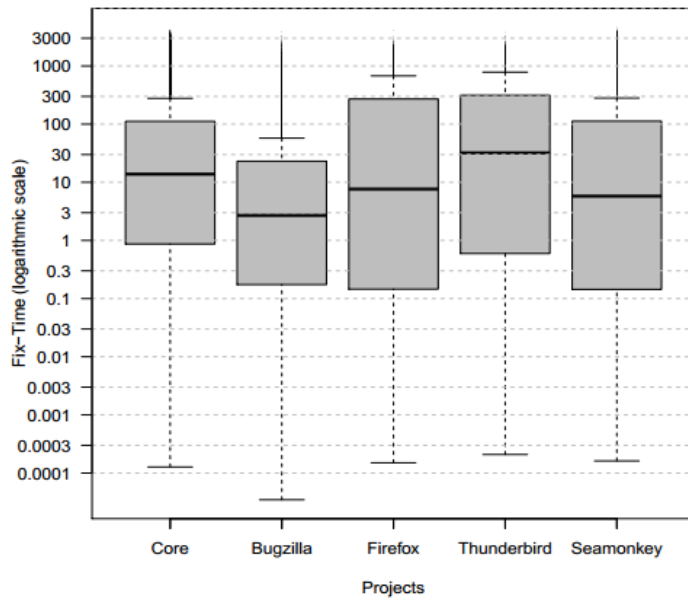


Figure 3.9 Boxplots of RT in days of Mozilla projects [21]

The authors decided to eliminate only those suspicious reports with very low resolution time and tended not to touch long-term reports. They proposed to set the RT threshold to half of the lower quartile of the RT distribution in order to eliminate suspicious reports, thus, the threshold is different for every project i.e. $\frac{1}{2} * Q_1$.

The same experiment as in [13] was conducted again in order to see the impact of the removal of outliers.

$$bugClass = \begin{cases} Fast : fixtime \leq median \\ Slow : fixtime > median \end{cases}$$

The formula above was used to classify issue reports by their RT. In order to classify the incoming bug, Naïve Bayes classifier was applied. Thus, the result before and after outlier removal is presented in Table 3.6.

(a) Accuracy Results of Eclipse Projects			(b) Accuracy Results of Mozilla Projects		
Project	AUC Before	AUC After	Project	AUC Before	AUC After
Eclipse Platform	0.692	0.700	Mozilla Core	0.663	0.686
Eclipse PDE	0.641	0.661	Mozilla Bugzilla	0.722	0.733
Eclipse JDT	0.646	0.649	Mozilla Firefox	0.623	0.653
Eclipse CDT	0.693	0.708	Mozilla Thunderbird	0.657	0.645
Eclipse GEF	0.663	0.732	Mozilla Seamonkey	0.698	0.706

Table 3.6 Accuracy before and after removal of outliers [21]

K-Fold cross-validation was used to assess prediction accuracy. For projects like Eclipse GEF, removal of outliers improves the accuracy rate for 0.069%. However, in case of Mozilla Thunderbird, the removal of outliers deteriorated the results.

The same study was extended in [22], where researchers tried several thresholds for eliminating outliers, including:

1. Half of the lower quartile: $\frac{1}{2} * Q_1$.
2. Median of the lower quartile.
3. Half of the upper quartile: $\frac{1}{2} * [Max - Q_3]$.
4. Median of the upper quartile.

5. Mild outliers of the upper inner fence, where inner fence is defined as $Q3 + 1.5 * IQ$, where IQ is inter-quartile.
6. Extreme outliers of the upper outer fence, where outer fence is defined as $Q3 + 3*IQ$, where IQ is the inter-quartile.

Inner and outer fence are described in Figure 3.10.

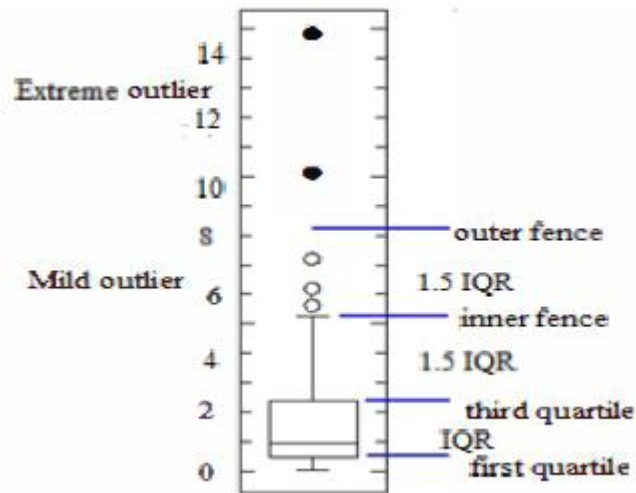


Figure 3.10 Outliers boundaries [22]

The study showed that eliminating outliers using thresholds for filtering out mild outliers produces the best results; classifying 71% of the issues correctly.

4 Application of Recommended Models to Fortumo Data

Since all suggested methods described in the previous section, were examined on different sets of data and measured using different accuracy measures, it is impossible to compare their performance.

In this section, we apply the suggested models from previous studies to Fortumo data in order to make their results comparable. We also measure prediction accuracy using the quality measures introduced in Section 2. Afterwards, we select the model with the highest prediction accuracy as the baseline model for further research.

In [13], the authors claimed that post-submission data improves prediction accuracy. Nevertheless, there is a measure of uncertainty in procuring a time estimate based on post-submission data for an organizations. Factors such as a large number of comments, developers involved in the project, and a huge volume of code that has been modified are more likely to extend the resolution time of an issue and people don't need any models to understand this phenomenon.

In this project, we focus on the initial data of an issue in order to make a prediction, because organizations need an RT estimate before resolution of the issue is completed.

Subsection 4.1 describes the data on which recommended models will be applied. Succeeding subsections are dedicated to the models themselves including:

1. K-Nearest Neighbors.
2. Naïve Bayes Classifier.
3. C4.5 Decision Tree.
4. Random Forest.
5. Ordinal Logistic Regression.

4.1 Issue Report Description

This section describes the process of issue report extraction from a bug-tracking system and the rules of their selection for this study. Moreover, it describes the content of an issue report and all its attributes that were used in the research.

Issue Reports Extraction and Selection

The data for the current study was taken from the JIRA bug tracking system of Fortumo. The data was extracted using an API provided by JIRA.

The selection of issue reports in this study was performed in the following order:

1. Separation of issues in English.

Previously, most issues were stored in Estonian. However, since two years ago, all the issues of the company have been stored in English language. Thus, some initial separation was applied and only the issues in English language were taken into account for in the context of this study. For separation, an existing library for R “textcat” [23] was used along with further manual double-checking of issues list.

2. Extracting issues with status “Closed”.

This study was conducted only on completed issues, in order to avoid cases with a partially tracked resolution time.

3. Extracting issues with defined resolution time.

The RT of most issues in the bug tracking system of Fortumo is tracked using Toggl [24]. These coverage of issues start mainly from the last 2 years.

Issue Report Attributes Description

In order to better understand the kind of data on which the following models would be applied, this subsection provides all the necessary information required.

Every issue used for our study is structured in the following way:

Attribute	Type	Values
Title	Text	
Description	Text	
Reporter	ENUM	70 different values
Project name	ENUM	11 different values
Type	ENUM	Bug, Epic, Gw-issue, Improvement, Incident, Investigation, New Feature, Project, Story, Sub-task , Task, Technical task

Priority	ENUM	Blocker, Critical, High, Immediate, Low, Normal
Creation date	Integer	Continuous values April 2011 – January 2015
Labels	Array of strings	39 different values
Resolution Time	Integer (seconds)	Recorded amount of spent time
Time Estimation	Integer (seconds)	Estimated amount of time, required for the issue

Table 4.1 Issue attributes

The extracted data contains the *Assignee* attribute which is not very useful because it is not static during an issue report lifecycle. In detail, first it takes a reporter of an issue as a value, then the concerned developer, the reviewer, release manager, and finally, back to the reporter. Consequently, assignees of the majority of closed issues are its reporters and as a result, it does not produce any additional value for our model.

The *Status* attribute is always closed since it is one of our issue filtering conditions. Consequently, this attributes doesn't produce any additional value for our model as well.

Fortunately, Fortumo's data possesses an attribute *Resolution Time* which describes an exact amount of time spent on an issue. According to the internal management, all developers always tracked the exact time they spent in completing the task.

The data was extracted in January 2015.

4.2 K-Nearest Neighbors

In order to repeat the approach described in [10], using the kNN modeling approach, we needed to follow the rules of Apache Lucene Text Similarity Engine, which was used in the study.

Thus, the following sequence of steps was performed:

1. Since the authors of [10] used *Lucene* as text similarity engine [25], we simulate this engine, executing the following steps:
 - a. Performing text preprocessing (details are presented in Section 5.3).

- b. Building Document-Term matrix with weights Tfidf (details are presented in Section 5.3).
 - c. Using Cosine Similarity as distance function for text.
2. For kNN algorithm: We applied kNN algorithm with k equals to 1, 3, 5, 9, separately for *issue description* and *issue title*, using the cosine similarity measure [26].
 3. For α -kNN algorithm: We applied α -kNN algorithm with α in 0.05, 0.1, 0.2, 0.3, 0.5 and 0.7 independently to *issue description* and *issue title*, using cosine similarity measure.
 4. We calculated the mean RT for k selected issue reports using *description-based* kNN.
 5. We calculated the mean RT for k selected issue reports using *title-based* kNN.
 6. We computed mean RT for values retrieved in 4) and 5) above.
 7. We transformed the result to the discrete scale of classes of RT.

The results of simulating the original study [10], is described in Figure 4.1 and Figure 4.2 where the choice of k (1, 3, 5 or 9) corresponds to that in the original paper.

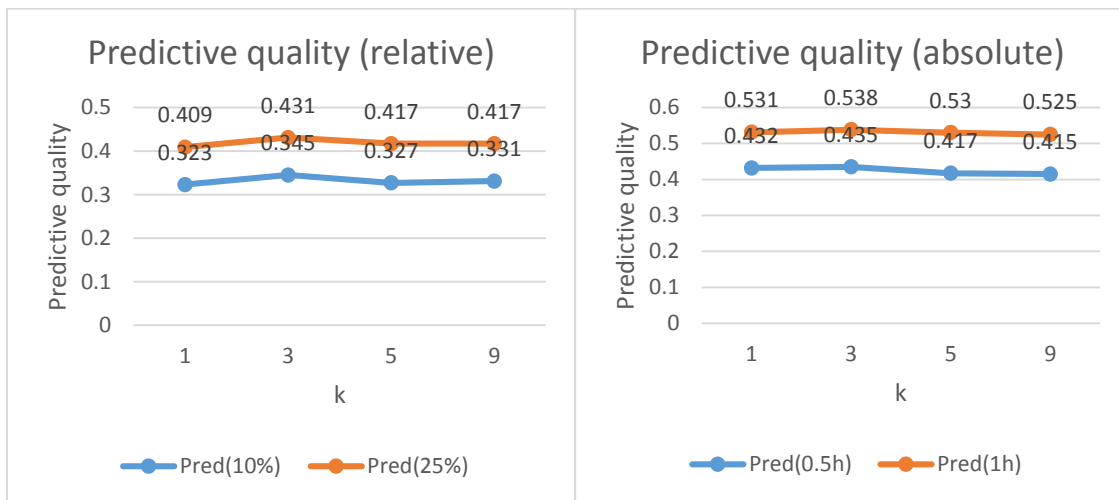


Figure 4.1 kNN approach results

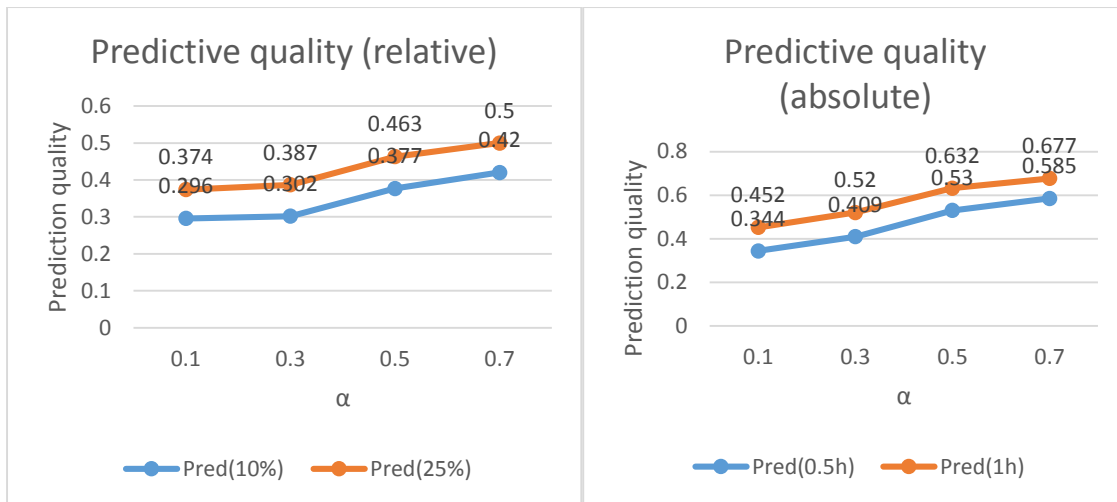


Figure 4.2 α -kNN approach results

α	Prediction rate
0.1	98.9%
0.3	72.9%
0.5	25.8%
0.7	8.4%

Table 4.2 α -kNN approach, prediction rate

The Prediction Rate is the percentage of issues which received a prediction.

It is worth mentioning that the cosine similarity measure returns values in range [0; 1]. When $\alpha=0.1$, some set of issues that do not receive any prediction. This occurs if the issue contains a very small set of words which are rarely used.

In this study [10], varying of k for kNN did not show any significant difference. Similarly, α -kNN in the same study shows the same tendency; the higher α , the higher the accuracy of prediction. α -kNN with $\alpha=0.7$ yields the best predictive quality however, accordingly to Table 4.2, only 8.4% of issue reports receive the prediction. In this thesis we assume that models with $\alpha > 0.3$ are useless for business purposes, since they don't return RT estimate in more than 90% of the cases.

Incidentally, from applying both methods on Fortumo data one can infer that kNN and α -kNN have an accuracy approximately 2.5 times and between 1.5-3.5 times higher than the original study, respectively.

We conclude that kNN, where $k=3$, is the best option according to Figure 4.1 and Figure 4.2 since it delivers the highest prediction accuracy and is the most useful for business purposes.

4.3 Naïve Bayes Classifier

Applying the Naïve-Bayes classifier studied in [16], and applied on Fortumo data, produces results which are described in Table 4.3. All the available attributes of the issue were passed as inputs to the classifier, namely:

1. Reporter.
2. Date.
3. Type.
4. Priority.
5. Project Name.
6. Labels.

Labels of an issue report are assigned to all issue reports as a Boolean flag. According to the table below, Naïve Bayes yields 12% of predictions with Relative Error of 10%.

Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)
0.138	0.171	0.121	0.138

Table 4.3 Predictive Quality of Naive Bayes algorithm

4.4 C4.5 Decision Tree

The C4.5 algorithm [27], applied on the same set of issue attributes, produces better results than Naïve Bayes because, as noted by other existing studies, C4.5 outperformed Naïve Bayes. The results in Table 4.4 show that C4.5 produces 27% of predictions have a Relative Error in the range of $\pm 10\%$ of the actual value and 50% of issue reports receive their prediction with Absolute Error of less than 1 hour.

Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)
0.460	0.566	0.378	0.439

Table 4.4 Predictive quality of the C4.5 algorithm

4.5 Random Forest

Random Forest [28], applied on the same set of Fortumo data following the idea described in [17] (with number of trees – 100, number of variables, sampled as candidates for split $=\sqrt{M}$, where M is the number of issue report features), obtained better results than C4.5. The results are shown in Table 4.5.

Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)
0.533	0.643	0.439	0.512

Table 4.5 Predictive quality of Random Forest

4.6 Ordered Logistic Regression

Since our dependent variable is ordinal, instead of Logistic Regression proposed in [15], we used Ordered Logistic Regression [29] and applied it using the following attributes: *Type*, *Priority*, *Project Name*, *Reporter*, *Creation date* and *Labels*. We obtained the results which shows in Table 4.6.

Pred(0.5h)	Pred (1h)	Pred(10%)	Pred(25%)	Prediction Rate
0.561	0.665	0.429	0.512	97%

Table 4.6 Predictive Quality of Ordered Logistic Regression

If some variable in a new incoming issue report occurs for the first time, then the model is unable to make a prediction. This is why only 97% of all issue reports received an RT estimate.

4.7 Other Methods

We were not able to reproduce Self-Organizing Maps which have been studied in [14] since the input to the method was not fully described in the paper.

Similarly, we were not able to reproduce clustering, described in [11], because of lack of information about how the clustering is implemented in SAS Text Miner and which interactive input Raja provided to SAS Text Miner during her research.

4.8 Summary

Thus, the research conducted on Fortumo's data shows that meta-information about the issue (*Type, Priority, Project Name, Reporter, Created date, Labels*) can bring about a higher predictive quality than analyzing issue report *title* and *description*. However, in this thesis we try to improve both types of models: text-based model and meta-information-based model.

Method	Pred(0.5h)	Pred(10%)
Best kNN (k=3)	0.435 AP: -35%	0.345 AP: -31%
Best α -kNN ($\alpha=0.3$)	0.409 AP: -39%	0.302 AP: -48%
Naïve Bayes Classifier	0.138 AP: -79%	0.121 AP: -76%
C4.5 decision tree	0.460 AP: -31%	0.439 AP: -12%
Random Forest	0.533 AP: -20.2%	0.439 AP: -12.4%
Ordered Logistic Regression	0.561 AP: -16%	0.429 AP: -16%

Table 4.7 Proposed models summary results

Table 4.7 summarizes the results of the various models applied to Fortumo data with regards to prediction quality, using one absolute and one relative quality measure. AP is the

abbreviation for ‘Actual Prediction Quality’ and is defined as the relative increase or decrease of prediction quality when comparing the proposed models to the current expert-based estimation practice at Fortumo.

According to Table 4.7, the accuracy of the proposed models is lower than the current accuracy of estimates in Fortumo. Consequently, one cannot perceive any benefit from using it. The aim of this thesis is to improve the proposed model, so that its accuracy will be higher than the current quality of RT predictions in Fortumo.

According to the Table 4.7, Random Forest [28], Ordered Logistic Regression [29] and kNN yield the best results.

The proposed methods listed in Table 4.7 can be roughly categorized into two main classes:

1. Meta-information-based model (Naïve Bayes Classifier, OLR, RF, C4.5 decision tree).
2. Text-based model (kNN and α -kNN).

The next step in our research is to select the best-performing model of each category and enhance their prediction quality using techniques described in Section 5.

Taking the best performing models in each category, we can define a baseline prediction accuracy as shown in Table 4.8.

	Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)
Meta information based model	0.561	0.665	0.439	0.512
Text based model	0.435	0.538	0.345	0.431

Table 4.8 Baseline Prediction Accuracy

For the meta-information-based model category we chose Random Forest and Ordered Logistic Regression, for the text-based model category we chose kNN with $k=3$ (as this choice of k yielded the best performance).

5 Case Study Elements

It is possible to divide the proposed models described in the previous section, into 2 categories:

1. Models using issue report meta-information: creation date, reporter, type of the issue, project, priority of the issue, etc.
2. Models using textual data: title and description.

We believe that it is possible to improve the accuracy of the recommended models. Consequently, in our study, we try to enhance the input data of the model and in the case of textual-based models, to improve the model itself.

One significant modification in our study from recommended models is applying the Moving Window concept which involves only the last part of issue reports as input data to the model. The motivation is described in the first part of this section. Then we describe the details of the meta-information-based and text-based models, respectively.

Hence, this section consists of the following components:

1. Moving Window Concept.
2. Meta-Information-Based Model.
3. Text-Based Model.

5.1 Moving Window Concept

In order to make our model work better, we decided to examine how the distribution of actual resolution times or distribution of actual RTs changes over time:

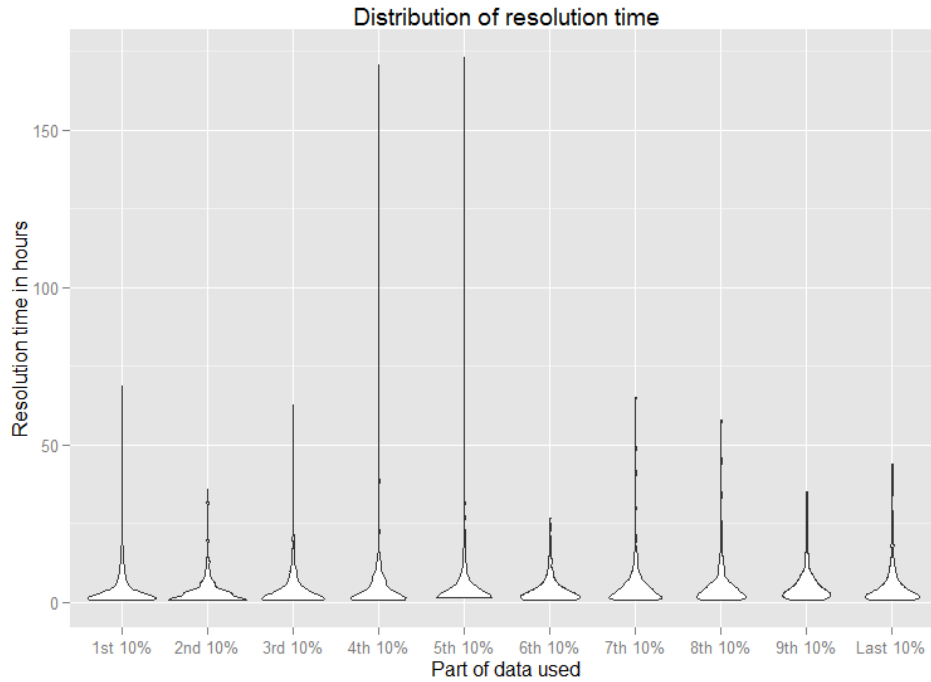


Figure 5.1 Distribution of actual resolution times over time

The figure above shows that the distribution of issues' RTs does change over time. It might happen along with changes in management, development process or other factors. Thus, in order to achieve better prediction accuracy, we decided to involve only the most recent set of issue reports to the prediction process. Namely, we involve 50 or 200 last issue reports.

5.2 Meta-Information-Based Model

According to the results presented in Section 4, Random Forest and Ordered Logistic Regression had the best performances. Due to this fact, we will base our case study on these classification approaches.

Firstly, following the ideas in other research studies that claimed that some features have a different degree of influence on RT than others, we will perform feature selection for improving input data to the model.

Secondly, as one of the recommended steps for improving the model described in Section 3.3, eliminating outliers might increase the quality of estimates. Thus, we will eliminate mild outliers in order to achieve better results.

Finally, we apply the Moving Window concept, which concerns involving only the most recent set of the data as the model input in order to improve prediction quality.

As a result, the case study contains the following steps:

1. Perform feature selection.
2. Removal of mild outliers from the issues.
3. Involving the Moving window concept in defining model input.

Feature Selection

We will examine which features of issue reports have a direct influence on resolution time. Depending on feature type, we perform the following set of tests on the feature and the resolution time in order to evaluate their connection and select only the most important issue report attributes as prediction model input:

1. Kruskal-Wallis test.
2. Chi-Square test.
3. Spearman correlation.

Removing Mild Outliers

According to [22] which was described in Section 3, removing *mild outliers of the upper inner fence* namely $Q3 + 1.5 * IQ$ where IQ is inter-quartile, brings about a higher predictive quality in comparison with other kinds of outliers as well as with that of no outlier removal. Thus, we will apply the aforementioned removal of outliers for our study.

5.3 Text Based Model

The authors of [10] proposed a model based on textual data of an issue report that applies kNN and α -kNN on the data. The need to know k in advance is the essential shortcoming of this modelling approach. However, the proposed workaround with using α was not successful since the rise of prediction quality was accompanied with a decrease in the number of predictions. In [11] it was proposed to use clustering for RT prediction. With kNN, it is a challenge to find an empirical way to define k and so we decide to use spherical k-means, as one of the well-known clustering techniques, instead of kNN in current thesis. K-means will construct the clusters with all maximally related issue reports together and the number of these related issue reports will no longer be a problem.

The fundamental concept of improving k-means in this thesis is dynamically defining an optimal k on every step which produces clusters of the best quality. Silhouette index will operate as a clustering quality measure. On each step, we find a possible range of optimal k ,

perform the clustering and define the final best k using Silhouette Index as a measure for calculating quality of a clustering.

Furthermore, we apply Latent Semantic Analysis (LSA) on textual data in order to create a semantic space of higher quality and overcome problems of polysemy and synonymy.

As we do with meta-information-based models we will again use the Moving Window idea to define model input.

Thus, the case study plan for text-based models contains the following steps:

1. Preprocessing textual data.
2. Applying Latent Semantic Analysis on textual data.
3. Selecting input data, using Moving Window concept.
4. Removing mild outliers from the issues.
5. Performing spherical k-means on data, while finding dynamically optimal k on each step using Silhouette index.

Preprocessing Textual Data

Text preprocessing includes the following steps:

1. Lowercasing the text.
2. Removing numbers from text.
3. Removing all punctuation from text.
4. Removing excessive whitespaces.
5. Removing stop words.
6. Applying Porter Stemming [30].
7. Transform corpora to Document-Term Matrix.
8. Applying Latent Semantic Indexing.

In order to use text information in our study, we structure our documents in the form of vector-space-based Term-Document Matrix. It is a common representation of document corpus, where terms are rows and documents are columns. Moreover, we use the TF-IDF matrix representation, which normalizes term frequency of every word using inverse document frequency (IDF). As a result of term frequency normalization, the weight and importance of commonly used terms throughout the document corpus is reduced, thus ensuring that document comparison will be more influenced by more discriminative words that rarely occur [31].

Calculating the Distance Between Documents

Term frequency: This refers to the number of occurrences of a term in a document divided by number of all words in a document:

$$tf(t, d) = \frac{n_i}{\sum_k n_k}$$

Inverse document frequency (IDF): This reduces the weight of commonly used words in the range of a particular set of documents. Every unique term in the current set of documents can have only one IDF value which is calculated as the number of all documents divided by the number of documents:

$$idf(t, D) = \frac{|D|}{|d_i \ni t|}$$

Where D is the document set, d_i is a document, t is term.

The TF-IDF value is calculated as:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

TF-IDF is used for building Term-Document Matrix for our model.

Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an automated mathematical technique which infers and extracts latent patterns in relationships between words or concepts that are applied to corpus of unstructured text. The LSA approach helps to overcome problems like synonymy and polysemy in text since it assumes that words with similar meaning occur in similar contexts.

The LSA technique uses Singular Value Decomposition (SVD) or Principle Component Analysis (PCA) in order to create a semantic space and reduce the dimensionality [32]. Dimensionality reduction exempts data from noise and thereby prepares better data for similarity-based data mining techniques like clustering. In addition, the removal of noisy dimensions helps to increase the importance of semantically significant data [31]. Hence, it is one of the most commonly used techniques for building semantic space and for further studies of the corpora.

In [33], LSA has been combined successfully with the Cosine similarity measure as a distance measure between documents for fuzzy c-means clustering bringing a much higher quality of clustering than in situations where LSA is not applied.

Additionally, LSA is not outperformed by other proposed methods for building semantic space [34]. LSA was successfully used in [11] where resulting data was clustered producing clusters with a significantly different mean.

Removing Mild Outliers

According to [22] which was described in Section 3, removing *mild outliers of the upper inner fence* namely $Q3 + 1.5 * IQ$ where IQ is inter-quartile, brings about a higher predictive quality in comparison with other kinds of outliers as well as with that of no outlier removal. Thus, we will apply the aforementioned removal of outliers in our case study.

Improved Spherical K-means Clustering

Since the title and description of an issue hold the majority of the issue information, we will try to involve it in the predictive model following ideas of [10]. However, unlike [10] we will cluster issues by its description and title using spherical k-means clustering with a dynamically tuned k.

Clustering documents is an important problem in text mining. The aim of it is to assign an appropriate label to each document and find the meaningful cluster centers. Clustering documents is used in other areas of text mining such as text categorization and information retrieval in which the labeled documents are needed.

K-means is one of the most popular unsupervised learning clustering algorithms. K-means algorithm works fast, is able to cluster several types of data including images, texts and others, and has a clear idea [31].

We decided to use k-means as an alternative to k-Nearest Neighbors used in [10] so we do not need to know how many close issues may exist for an incoming issue.

In [35], it was shown that k-means outperforms fuzzy c-means clustering when the dataset is big and realistically noisy.

However, as with other approaches, k-means has its own weaknesses. A major weakness is that the user of the algorithm must define k, the number of clusters to which documents should be separated. Since in our case, it is impossible to have k predefined because we never know how many issue topics are actually covered in a given set of issue reports, we propose a method which helps us overcome current vulnerability. It will consist of the following components:

1. Predict possible $k_{optimal}$: For the first issue we set $k_{optimal}$ as the number of existing projects in the dataset. Otherwise, we set $k_{optimal}$ as previous best k . Since the number of existing projects in Fortumo dataset is 11, in this thesis we assume that this initial value is sufficient enough for the first prediction. However, additional research should be conducted in order to define the initial $k_{optimal}$. Because of time constraints for this thesis, we don't perform such research.
2. Next, we define the range of optimal k as $k_{optimal} \pm 2$. We assume that k must not be critically different between two subsequent steps, so the margin ± 2 should suffice to find the best clustering and preserve reasonable speed of performance of an algorithm. Additionally, this margin must be sufficient enough to reach the best k during small amount of steps.
3. We perform the clustering for every k in a predefined range. Finally, we calculate the quality of each clustering using Silhouette Index and select the best k .

Cosine Distance and Spherical K-means

Lucene Apache Text Similarity Engine [36] involved in [10] uses Cosine distance for text clustering. We follow the same ideas, since [37] outlines a better performance of the *cosine similarity measure* applied on large document corpus over the set of measures like *neighborhood similarity*, *shortest path*, *neighborhood with features*, *fail distance* and *voltage based similarity measure*. Additionally, another study [38] showed that classical k-means with Euclidean distance yields poor results when spherical k-means usually outperforms it.

Let \vec{u} and \vec{v} be vectors of same length of Term-Document Matrix which represents vectors of terms. The cosine distance between vectors (an angle) is defined as follows [31]:

$$\cos(\vec{u}, \vec{v}) = \frac{\vec{u} * \vec{v}}{|\vec{u}| |\vec{v}|} = \frac{\sum_i u_i * v_i}{\sqrt{\sum_i u_i^2 * \sum_i v_i^2}}$$

Silhouette Index

Silhouette index is a measure often used for measuring cluster quality, which is defined in the following way:

Let us consider a measure which calculates the average distance between the element and all its neighbors in a cluster:

$$a(i) = \frac{1}{n} \left(\sum_{j=1, i \neq j}^n \text{distance}(c_i, c_j) \right)$$

Where n is the number of elements in a cluster C_i and $c \in C_i$.

The distance between an element and another cluster is the smallest distance between itself and all other elements of another cluster:

$$\text{dist}(c, C_j) = \min \text{distance}(c, c_{ij})$$

Let us consider a measure to calculate the smallest distance between the element and all other clusters:

$$b(i) = \min \text{dist}(c_i, C_j)$$

Then the Silhouette Index of i cluster is defined as follows:

$$SI(C_i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Overall Silhouette Index of the whole clustering will be:

$$SI = \text{avg}(SI(C_i)) \text{ [39]}$$

Cluster Predictor

Cluster predictor is defined as the *median* of the RTs of all issues contained in a cluster. Median is used since the RT distribution is skewed.

6 Case Study Execution and Results

This section is dedicated to presenting and discussing the results of the approaches suggested in Section 5 when applied to Fortumo’s data. Similar to Section 5, this section is divided into 2 parts:

1. Enhancement of Accuracy of Meta-Information-Based Model Prediction.
2. Enhancement of Accuracy of Text-Based Model Prediction.

We didn’t benefit with meta-information-based Random Forest and Ordered Logistic Regression. However, we received higher accuracy with enhanced text-based model, having 34% of improvement from baseline model and up to 20% from an actual prediction accuracy.

The description of issue reports have already been presented in the Section 4.1.

6.1 Enhancement of Accuracy of Meta-Information Based Model Prediction

In this section, according to the Section 5.2, we perform the described enhancement techniques on Random Forest and Ordered Logistic Regression.

Firstly, following the ideas in other research studies that claimed that some features have a different degree of influence on RT than others, we perform feature selection for improving input data to the model.

Secondly, we will eliminate outliers in order to increase the quality of estimates.

Finally, we apply the Moving Window concept, which concerns involving only the most recent set of the data as the model input in order to improve prediction quality.

Feature Selection

Using the Kruskal-Wallis Test, we examined the dependency of resolution time on such issue report attributes of ordinal type like *median RT of reporter*, *median RT of issue project*, *median RT of issue type*, *median RT of issue priority*, as shown in Table 6.1.

Attribute	Kruskal-Wallis chi-squared	p-value	df
Median RT of reporter	165.86	< 2.2e-16	7

Median RT of issues in given project	231.25	<2.2e-16	5
Median RT of issues of given type	65.14	4.041e-12	6
Median RT of issues of given priority	3.05	0.384	3

Table 6.1 Kruskal-Wallis test between attributes and resolution time

Additionally, we applied the Spearman correlation on the set of issue attributes of continuous type, as shown in Table 6.2.

Attribute	Correlation
Number of issues of given priority	0.12
Number of issues of a given project	-0.11
Number of issues of a given type	0.00
Number of issues of a given reporter	0.12
Date	0.145
Year	0.074

Table 6.2 Spearman correlations between attributes and resolution time

Attribute	X-squared	p-value	df
Reporter	896.580	< 2.2e-16	483
Priority	31.399	0.643	35
Type	222.109	5.322e-16	77
Project Name	308.850	< 2.2e-16	70

Table 6.3 Chi-square test results between attributes and resolution time

According to the results shown in Table 6.1, Table 6.2 and Table 6.3, we exclude the attributes *Priority* and *Median RT of issue of a given priority* from the model as resolution time does not seem to depend on this data. In addition, all attributes mentioned in Table 6.2 would not be included in the model due to the fact that there is no correlation between them and RT.

We also examined how the Kruskal-Wallis Test results and Spearman correlations change if we calculate such issue report attributes as: *Number of issues of given priority*, *Number of issues of given type*, *Number of issues of given project*, *Number of issues of given reporter* and *Median RT of given priority*, *Median RT of given type*, *Median RT of given project*, *Median RT of given reporter* using only the most recent 50 issues. According to the results described in Table 6.4 and Table 6.5, the tendency remains unchanged.

Attribute	Number of issues	Kruskal-Wallis chi-squared	P-value
Median RT of given reporter	All issues	165.86	< 2.2e-16
	50	97.885	< 2.2e-16
Median RT of given type	All issues	65.14	4.041e-12
	50	48.612	2.702e-08
Median RT of given priority	All issues	3.05	0.384
	50	5.466	0.362
Median RT of given project	All issues	231.25	< 2.2e-16
	50	164.195	< 2.2e-16

Table 6.4 Kruskal-Wallis test results between attributes (calculated with Moving Window) and resolution time

Attribute	Correlation
Number of issues of given priority	0.05
Number of issues of a given project	-0.23
Number of issues of a given type	-0.14

Number of issues of a given reporter	0.04
--------------------------------------	------

Table 6.5 Spearman correlation between attributes (calculated with Moving Window) and the resolution time

Another attribute of an issue report is custom *label*. Every issue can have any number of labels, which user creates by himself. Currently, in the repository, dataset has 39 defined labels:

Label	Number of issues	Kruskall-Wallis chi-squared	P value
Gw-dev	8	3.358	0.067
Integrations	26	2.689	0.101
Front-end	13	0.016	0.899
Manual-work	257	54.626	1.458e-13
Operations	1	0.937	0.333
Vc-calculations-errors	2	0.148	0.701
Integrations	4	2.959	0.085
Wutlar	1	0.516	0.473
Msggrooming20141208	10	0.898	0.343
Subsonic	1	1.620	0.203
Penny	1	2.453	0.117
Centili	2	1.982	0.156
NTH	1	2.906	0.088
100/30	1	2.249	0.134
Greece	1	2.249	0.134
M-stat	1	2.249	0.134
Inapp	1	1.620	0.203

Integrat	27	3.057	0.080
Telkom	1	0.515	0.473
Compliance	9	0.386	0.534
Dcb	1	1.620	0.203
Verse	2	1.982	0.159
penny	1	2.453	0.117
Purser	3	3e-04	0.986
Judge	1	1.620	0.203
Outofsprint	2	0.309	0.580
Spendinglimits	1	0.516	0.473
RZA	1	0.516	0.473
Spain	1	0.030	0.861
Timwe	1	0.516	0.473
timwe	1	0.304	0.861
Technicaldebt	9	0.466	0.496
Documentation	1	0.030	0.862
US	1	2.906	0.089
Disney	3	0.706	0.401
Indosat	1	0.516	0.473
Msggroomin20150105	5	2.519	0.113
Msggrooming20150119	1	0.516	0.473
Recalculate	3	1.485	0.223

Table 6.6 Kruskal-Wallis test results for custom label entries

	Number of issues	Kruskal-Wallis chi-squared	R value
--	------------------	----------------------------	---------

No labels	1764	20.030	7.607e-06
-----------	------	--------	-----------

Table 6.7 Kruskal-Wallis test results for issues without custom label entry

According to the Table 6.6 and Table 6.7, labels were not used on a large scale since the majority of them were attached only to one issue. However, the label *Manual Work* which was the most-often used label depicts an influence on issue RT. In addition, it is possible to say the same about issues that are not labeled (see Table 6.7). Consequently, using the *Manual Work* label or *No Labels* attribute might improve future model performance.

Model Application Results

Considering that we currently possess more knowledge about which attributes influence the RT, we apply Random Forest to only significant ones.

Since Random Forest does not handle missing values which definitely occur in attributes such as *Average RT for reporter/type/project* (when the first type reporter/type/project comes in), we decided to just eliminate issues with missing attributes from the model. Such issues might occur often in the beginning, but their amount decreases over time. Eventually, 2-3% of issue reports did not receive prediction because of missing values.

Table 6.8 presents results of performing Random Forest. The results shown in Table 6.8 indicate that there is only a small improvement over the baseline performance, if all issue reports are used for prediction. However, the improvement is too small to outperform the current expert-based prediction quality at Fortumo.

N. of issues	Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)	Prediction Rate
Last 50	0.500 BP: -11.3% AP: -25.1%	0.605 BP: -9% AP: -16.8%	0.408 BP: -7% AP: -18.6%	0.472 BP: -7.7% AP: -18.3%	97%
Last 200	0.536 BP: -4.5% AP: -19.8%	0.644 BP: -3.2% AP: -11.4%	0.435 BP: -0.9% AP: -13.1%	0.513 BP: -0.1% AP: -11.3%	98%

All	0.575	0.668	0.453	0.538	97%
	BP: +2.5%	BP: +0.4%	BP: +3.2%	BP: +5.1%	
	AP: -13.9%	AP: -8.2%	AP: -9.6%	AP: -6.9%	

Table 6.8 Prediction quality of RF with feature selection

Table 6.9 shows the results of performing Ordered Logistic Regression on Fortumo data. BP shows the relative improvement of prediction quality as compared to the baseline, and AP shows the relative difference between the performance of the k-means approach as compared to the currently used expert-based approach at Fortumo.

Unfortunately, in all cases the model yields worse results than the baseline prediction model.

N. of issues	Pred(0.5h)	Pred(1h)	Pred(10%)	Pred(25%)	Prediction rate
Last 50	0.436	0.525	0.357	0.406	69%
	BP: -22.3%	BP: -21.0%	BP: -18.8%	BP: -20.7%	
	AP: -34.7%	AP: -27.8%	AP: -28.8%	AP: -29.8%	
Last 200	0.490	0.598	0.392	0.462	91%
	BP: -12.6%	BP: -10.1%	BP: -10.8%	BP: -9.8%	
	AP: -26.6%	AP: -17.7%	AP: -21.8%	AP: -20.1%	
All	0.524	0.621	0.416	0.497	95%
	BP: -6.6%	BP: -6.7%	BP: -5.3%	BP: -3%	
	AP: -21.5%	AP: -14.6%	AP: -17.0%	AP: -14.1%	

Table 6.9 Prediction quality of OLR with feature selection

Such results can be caused by the fact that 50 or 200 last issue reports does not give enough information for both Random Forest and Ordered Logistic Regression.

For logistic regression, both moving window and feature selection caused deterioration in the quality of predictions.

Using the method suggested in [22] for eliminating upper *mild outliers* such as $Q3 + 1.5IQ$, we performed this step for every prediction and received the results shown in Table 6.10 and Table 6.11.

N. of issues	Pred (0.5h)	Pred (1h)	Pred (10%)	Pred (25%)	Prediction rate
Last 50	0.500 BP: -10.8% AP: -25.1% WRO: 0	0.614 BP: -7.6% AP: -15.5% WRO: +1.5%	0.415 BP: -5.5% AP: -17.2% WRO: +1.7%	0.484 BP: -5.5% AP: -16.3% WRO: +2.5%	97%
Last 200	0.535 BP: -4.6% AP: -19.9% WRO: -0.2%	0.643 BP: -3.3% AP: -11.6% WRO: -0.2%	0.438 BP: -0.3% AP: -12.6% WRO: +0.7%	0.515 BP: -0.7% AP: -10.8% WRO: +0.4%	98%
All	0.572 BP: +2.0% AP: -14.3% WRO: -0.5%	0.666 BP: +0.1% AP: -8.4% WRO: -0.3%	0.449 BP: +2.3% AP: -10.4% WRO: -0.9%	0.535 BP: +4.5% AP: -7.4% WRO: -0.6%	97%

Table 6.10 Prediction quality of RF with feature selection and removal of outliers

N. of issues	Pred (0.5h)	Pred (1h)	Pred (10%)	Pred (25%)	Prediction rate
Last 50	0.460 BP: -18.0% AP: -31.1% WRO: +5.5%	0.566 BP: -14.9% AP: -22.2% WRO: + 7.8%	0.378 BP: -14.0% AP: -24.6% WRO: +5.9%	0.436 BP: -14.9% AP: -24.6% WRO: +7.4%	65%
Last 200	0.514 BP: -8.3% AP: -23.0% WRO: +4.9%	0.622 BP: -6.4% AP: -14.4% WRO: +4.0%	0.408 BP: -7.1% AP: -18.6% WRO: +4.1%	0.483 BP: -5.6% AP: -16.4% WRO: +4.5%	82%
All	0.579	0.676	0.457	0.541	86%

	BP: +3.2%	BP: +1.7%	BP: +4.1%	BP: +5.7%	
	AP: -13.3%	AP: -7.0%	AP: -8.8%	AP: -6.3%	
	WRO: + 10.5%	WRO: +8.9%	WRO: +9.9%	WRO: +8.9%	

Table 6.11 Prediction quality of OLR with feature selection and removal of outliers

In Table 6.10 and Table 6.11 WRO shows relative improvement of prediction quality as compared to the same model without the removal of outliers.

Since Ordered Logistic Regression cannot make a prediction for an incoming issue report in case some attribute value occurs for the first time, the percentage of issues, which receive a prediction, decreases especially in the case of moving window with a lesser number of issue reports.

According to results, described in Table 6.10 and Table 6.11, additional removal of outliers has not caused any improvement in comparison with the same method, applied without it.

6.2 Enhancement of Accuracy of Text-Based Model Prediction

This section of research is dedicated exclusively to the prediction of RT based on issue title and description. We ran spherical k-means after text preprocessing with k defined dynamically based on the cluster quality measure described earlier.

In this thesis, due to time constraints, we do not study this approach without Moving Window, i.e. with all previous data involved.

Predictive quality of current approach without the removal of outliers is described in Table 6.12. According to the results, models based on *title* and *description* produce approximately same prediction accuracy. In addition, using a lesser number of issues for prediction improves the results. In addition, Table 6.12 shows that the prediction accuracy of a given model is much better than the baseline accuracy and slightly better than the accuracy of expert-based estimates regarding relative error. However, it is slightly worse than the accuracy of expert estimates regarding absolute error.

N. of issues involved	Data clustered	Pred (0.5h)	Pred (1h)	Pred (10%)	Pred (25%)

Last 50	Title	0.643 BP: +47.8% AP: -3.7%	0.732 BP: +36.0% AP: +0.6%	0.603 BP: +74.7% AP: +20.3%	0.656 BP: +52.2% AP: +13.5%
Last 200		0.637 BP: +46.3% AP: -4.7%	0.722 BP: +34.2% AP: -0.7%	0.589 BP: +70.8% AP: +17.6%	0.650 BP: +50.7% AP: +12.4%
Last 50	Description	0.615 BP: +41.4% AP: -7.9%	0.703 BP: +30.6% AP: -3.3%	0.558 BP: +61.8% AP: +11.4%	0.612 BP: +41.9% AP: +5.8%
Last 200		0.606 BP: +39.3% AP: -9.3%	0.708 BP: +31.7% AP: -2.5%	0.556 BP: +61.2% AP: +11.0%	0.617 BP: +43.0% AP: +6.7%

Table 6.12 Predictive quality of improved text-based model without removal of outliers

In Table 6.12, BP shows the relative improvement of prediction quality as compared to the baseline, and AP shows the relative difference between the performance of the k-means approach as compared to the currently used expert-based approach at Fortumo.

Table 6.13 describes the general distribution of an optimal number of clusters found for every prediction. In general, title-based clustering generates more clusters than description-based clustering. Also, the more issue reports are involved for clustering, the more clusters the model generates.

N. of issues involved	Data clustered	Distribution of clusters number
-----------------------	----------------	---------------------------------

Last 50	Title	<p style="text-align: center;">Distribution of optimal number of clusters</p>
Last 200		<p style="text-align: center;">Distribution of number of clusters</p>
Last 50	Description	<p style="text-align: center;">Distribution of number of clusters</p>

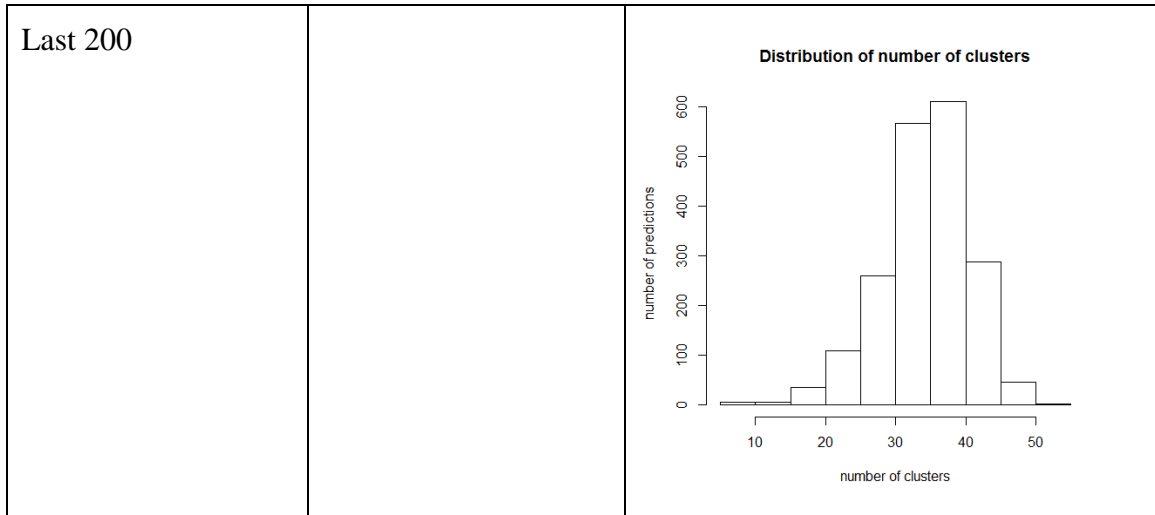
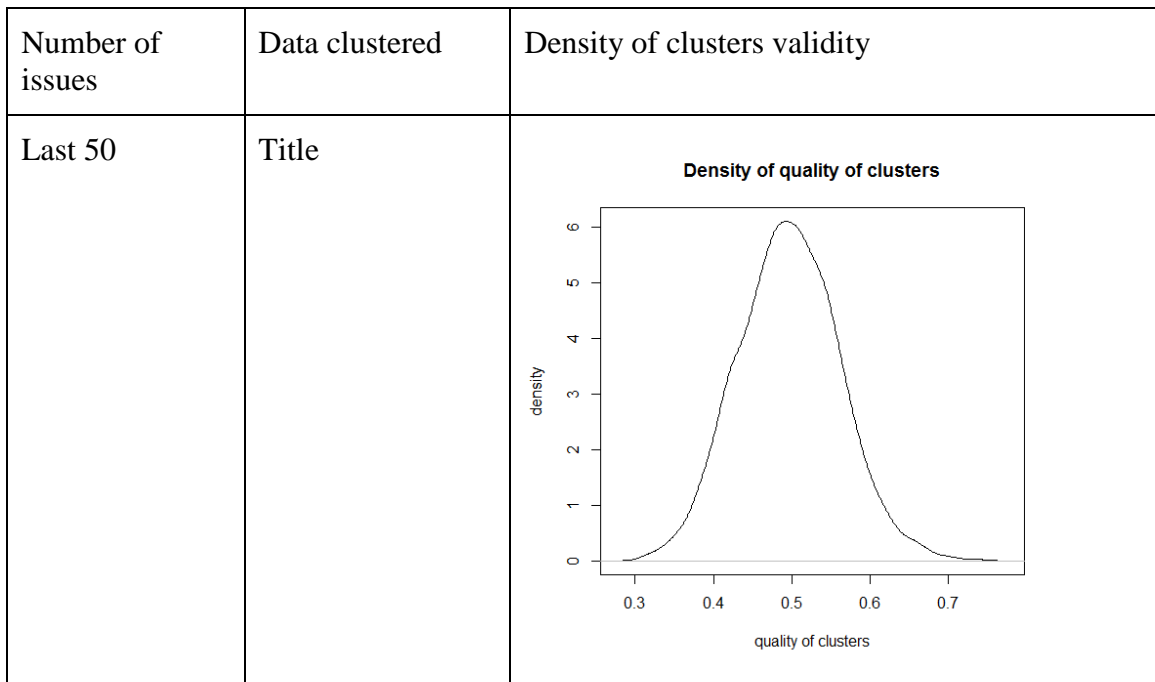


Table 6.13 The distribution of number of clusters defined on every step

Table 6.14 depicts the distribution of clustering quality measured on every prediction step. In general, the quality of title-based clustering exceeds that of description-based clustering. According to Table 6.12 and Table 6.13, the better clustering quality we have, the higher the prediction accuracy becomes. Thus, it might be the case that there is a correlation between the quality of clustering and prediction accuracy.



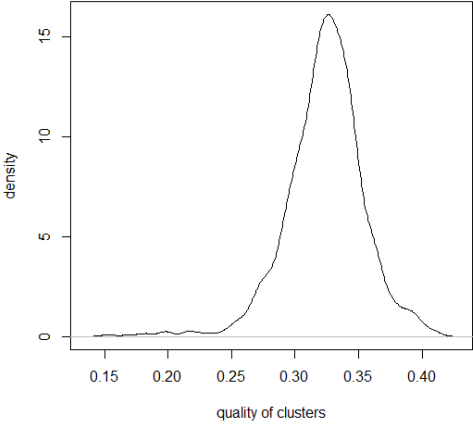
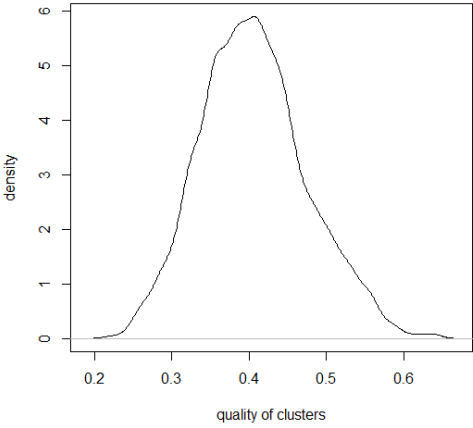
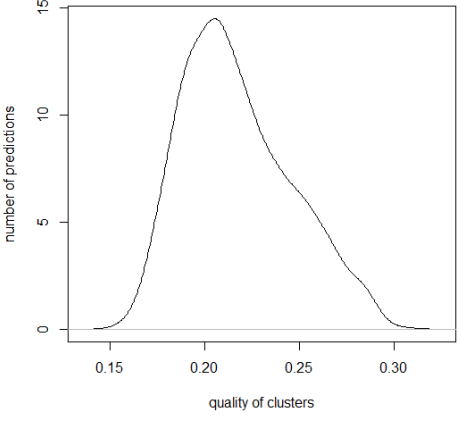
Last 200		<p style="text-align: center;">Density of quality of clusters</p> 
Last 50	Description	<p style="text-align: center;">Density of quality of clusters</p> 
Last 200		<p style="text-align: center;">Density of quality of clusters</p> 

Table 6.14 Density of quality of clusters

According to Table 6.14, the more issue reports we involve to the model, the worse the quality of clustering becomes. Thus, the model yields poor prediction accuracy. However, an average number of issue reports in a single cluster for both cases is approximately 3-4.

In order to analyze how the removal of outliers influences the results, we conducted the same experiment but with the removal of outliers. Table 6.15 described the results of this research. Table 6.15 depicts that the prediction accuracy of this approach is much better than the baseline accuracy. Moreover, it is slightly worse than current practice if to compare using absolute error, but slightly better than current practice if to compare using relative error. Unfortunately, the removal of outliers doesn't improve the accuracy.

N. of issues involved	Data clustered	Pred (0.5h)	Pred (1h)	Pred (10%)	Pred (25%)
Last 50	Title	0.627 BP: +44.1% AP: -6.2% WRO: -2.5%	0.715 BP: +33.0% AP: -1.6% WRO: -2.3%	0.583 BP: +69.1% AP: +16.4% WRO: -3.3%	0.638 BP: +48.1% AP: +10.4% WRO: -2.7%
Last 200		0.620 BP: +42.5% AP: -7.2% WRO: -2.7%	0.716 BP: +33.1% AP: -1.5% WRO: -0.8%	0.573 BP: +66.0% AP: +14.3% WRO: -2.7%	0.642 BP: +48.9% AP: +11% WRO: -1.2%
Last 50	Description	0.605 BP: +39.1% AP: -9.4% WRO: -1.6%	0.697 BP: +29.6% AP: -4.1% WRO: -0.9%	0.544 BP: +57.6% AP: +8.5% WRO: -2.5%	0.603 BP: +39.9% AP: +4.3% WRO: -1.5%
Last 200		0.601 BP: +38.1% AP: -10.1% WRO: -0.8%	0.695 BP: +29.2% AP: -4.4% WRO: -1.8%	0.550 BP: +59.5% AP: +9.8% WRO: -1.1%	0.613 BP: +42.3% AP: +6.1% WRO: -0.6%

Table 6.15 Prediction quality of improved text-based model with removed outliers

A more detailed description of how the removal of outliers influences the prediction quality is presented in Figure 6.1. As expected, removal of outliers caused a deterioration in the quality of prediction for issues with large RT, since the necessary previous data for them was eliminated from the input as outliers.

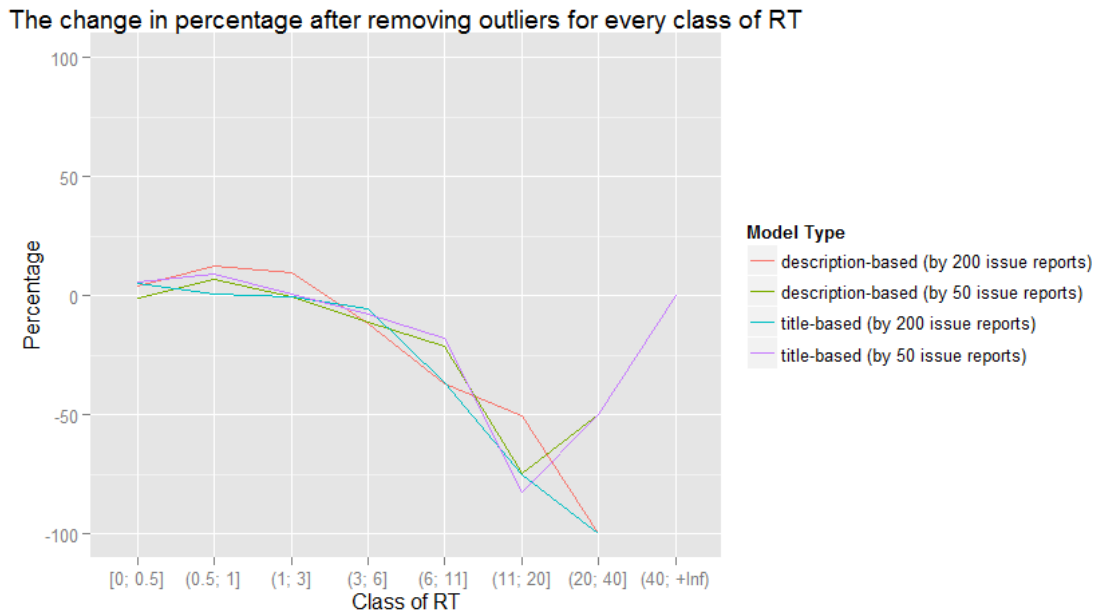


Figure 6.1 The change in percentage after removing outliers for every class of RT

6.3 Discussion

Based on the results presented in Sections 6.1 and 6.2, we saw that the improved text-based model could achieve better prediction quality than the currently used expert-based practice at Fortumo. This could not be achieved with the improved meta-information-based models (both RF and OLR).

We examined the distributions of predicted RT of best meta-information-based model (OLR without Moving Window and with removal of outliers, RF without Moving Window and without removal of outliers) and text-based model (title-based clustering with 50 last issue reports involved without the removal of outliers) in comparison with the distribution of actual RT. This is described in Figure 6.2. According to the figure meta-information-based models predict RT mostly to the (1; 3] interval, which is not close to the actual RT distribution. Also,

the distribution of prediction of the text-based model is much more similar to the actual RT distribution.

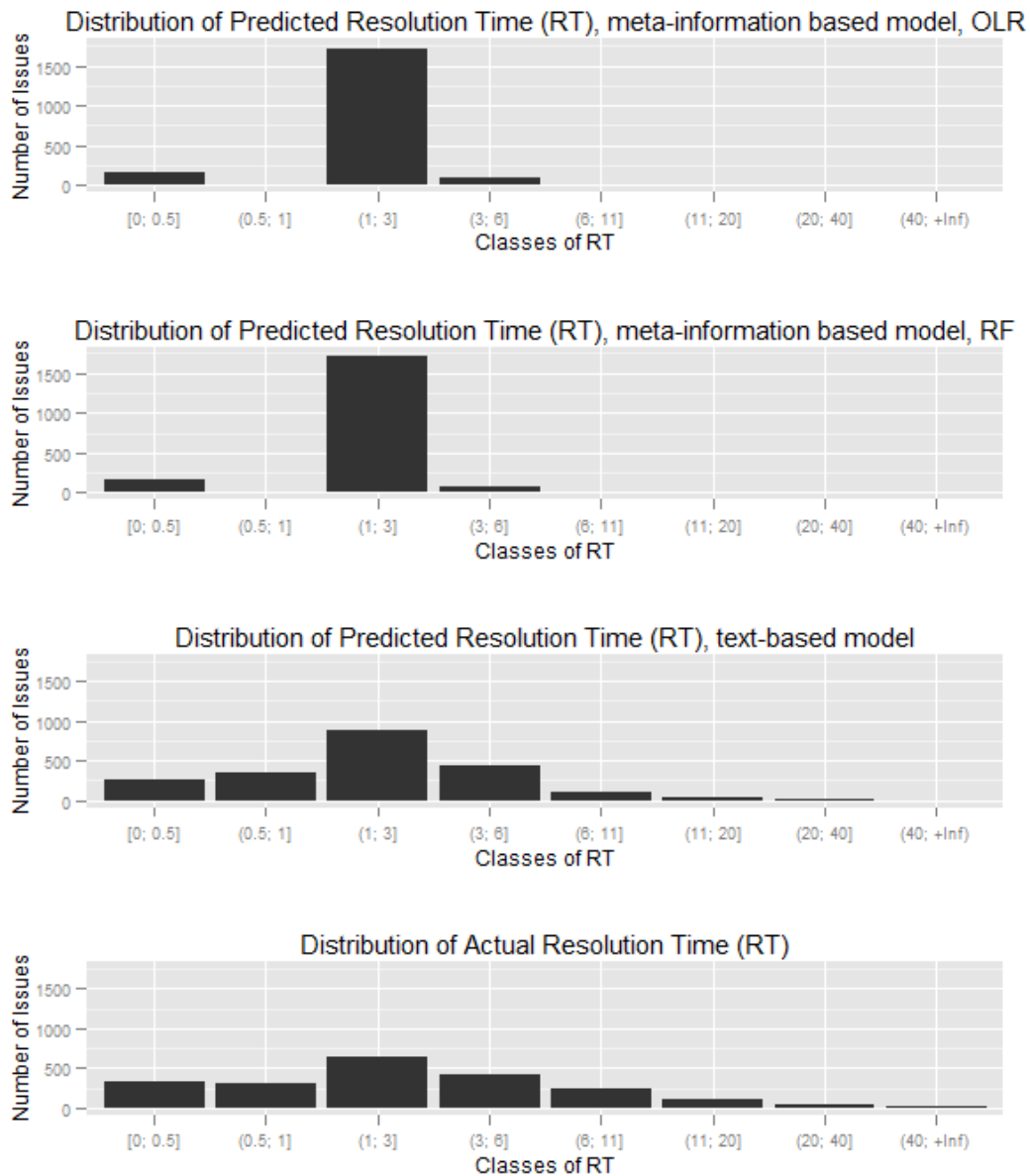


Figure 6.2 Comparison of distributions of predicted RT of best models with the distribution of actual RT

In contrast, the distribution of predicted RT by text-based model resembles the distribution of actual RT (Figure 6.3). Since we noticed earlier (Section 2) that the expert-based RT

predictions at Fortumo are generally over-optimistic, i.e., systematically underestimate the actual RT, we took a closer look at the differences between the distribution of RT predicted with the text-based model as compared to the distribution of actual RT values.

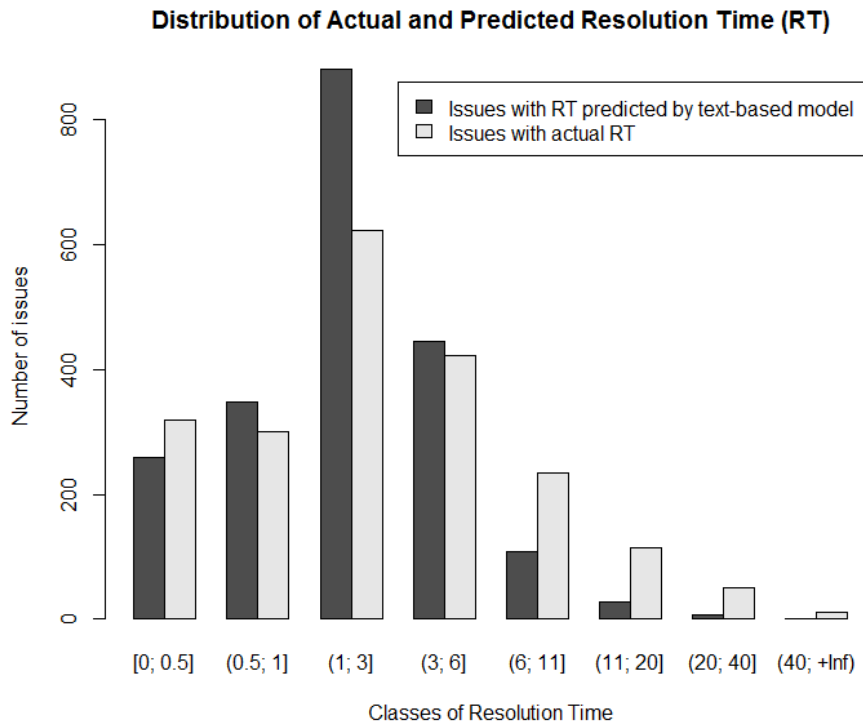


Figure 6.3 Comparison of predicted RT distribution of text based model in comparison with distribution of actual RT

According to Figure 6.3, the model still underestimates long-term issues (as well as experts) but predicts the RTs of issue reports with short RT more accurately than experts. Having the model, which has a prediction accuracy close to that of the experts' estimates has some advantages. One of these are that it is possible to replace experts with the model in case the expert is absent since it may be the case that personnel changes and trained people are replaced with new ones, the model might provide a decision-making support equivalent to experienced employees. In addition, we observed that experts are over-optimistic when dealing with issue reports having short RT – the model is more realistic. Thus, the model could help make expert-predictions more realistic, if experts use the model in addition to their expertise.

It is worth mentioning however, that in this thesis none of the methods used in the literature (and applied to Fortumo data – Section 4) could be improved to become better than the currently used expert-based approach. We assume that this is a good field for further investigation with regard future work.

In addition, Magne Jørgensen in his article [40] concludes that best software effort prediction model doesn't exist since the context and variables with the largest impact on the effort varies

between projects. Hence, the approach suggested in this study, should be tested on different data in order to evaluate their performance in different contexts.

Although the meta-information-based model did not yield high prediction accuracy, I believe that meta-information is still a carrier of important knowledge about an issue. With regards to future work, other prediction models based on meta-information are considered. Moreover, the combination of both types of models should be reviewed as well, since I believe that the combination might bring better results. Furthermore, Magne Jørgensen in [40] claims that the average of predictions from different sources are much more likely to be precise than a single estimate.

Text-based models without Moving Window has not been studied as well. Given our positive results with text-based models, we consider further improvement of these models another promising area for future research.

Finally, it is worth considering which data people possess and involve into the process of estimation of issue reports resolution time which has not been involved in the models in this thesis. It implies involving more data from other sources, like: code repository, pull requests data, projects documentations, projects notifications, etc.

7 Conclusion

In this thesis, we calculated the prediction quality of experts' estimates with regards to issue report resolution time based on data provided by Fortumo OÜ. In addition, we compared different models proposed in existing studies with respect to the quality of predictions concerning issue report resolution time, and found models with the highest accuracy of predictions. Subsequently, we defined the prediction accuracy of the best suggested models as the baseline accuracy.

We divided the succeeding study into two parts:

- 1) Enhancement of the accuracy of meta-information-based model.
- 2) Enhancement of the accuracy of text-based model.

Having Random Forest and Ordinal Logistic Regression as the best meta-information-based models, we applied a set of different techniques on input data in order to improve their prediction quality. However, we only achieved a tiny percentage of improvement.

We constructed a text-based model as an amalgamation of different existing approaches, having text-based clustering as a key concept; and achieved a better prediction accuracy than that of the experts' estimates. Such results introduce the possibility to replace experts with the model, should the need arise. Additionally, in our study, we conclude that there is a strong correlation between the quality of clustering and the accuracy of resolution time prediction.

Furthermore, according to the distribution of experts' estimates and predictions of the text-based model, both approaches have similar tendency to underestimate long-term issues. However, the text-based model, studied in this thesis, predicts RTs of issue reports with short RT more accurately than the experts. Such behavior implies the possibility of supplementing the expert's opinion. In addition, text-based models produced a higher prediction quality, approximately up to 20% better than estimates made by experts regarding relative error.

8 Bibliography

- [1] B. Hartman, "An Introduction to Planning Poker | Agile Zone," DZone, 11 11 2009. [Online]. Available: <http://agile.dzone.com/articles/introduction-planning-poker>. [Accessed 6 6 2015].
- [2] B. W. Boehm, *Software Engineering Economics*, Prentice Hall PTR Upper Saddle River, NJ, USA, 1981.
- [3] "COCOMO II - Constructive Cost Model," USC - Viterbi School of Engineering, 2014. [Online]. Available: <http://csse.usc.edu/tools/COCOMOII.php>. [Accessed 2 5 2015].
- [4] M. Madheswaran; D.Sivakumar, "Enhancement of Prediction Accuracy in COCOMO Model for Software Projects Using Neural Network," in *Computing, Communication and Networking Technologies (ICCCNT)*, Hefei, 2014.
- [5] Iman Attarzadeh; Amin Mehranzadeh; Ali Barati, "Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 IEEE International Conference*, Phuket, 2012.
- [6] Ratnesh Litoriya; Narendra Sharma; Dr. Abhay Kothari, "The Effects of Data Mining Techniques on Software Cost Estimation Effort using Agile COCOMO II," in *Engineering Management Conference, 2008. IEMC Europe 2008. IEEE International*, 2012.
- [7] Z. Dan, "Improving the Accuracy in Software Effort Estimation Using Artificial Neural Network Model Based on Particle Swarm Optimization," in *Service Operations and Logistics, and Informatics (SOLI), 2013 IEEE International Conference*, Dongguan, 2013.
- [8] Iman Attarzadeh; Siew Hock Ow, "Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference*, Taipei, 2011.

- [9] C. F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Communications of the ACM*, vol. 30, pp. 416-429, 1987.
- [10] Cathrin Weiss, Rahul Premraj, Thomas Zimmermann, Andreas Zeller, "How Long Will It Take to Fix This Bug," in *Mining Software Repositories, 2007. ICSE Workshops MSR '07, IEEE*, Minneapolis, MN, 2007.
- [11] U. Raja, "All complaints are not created equal: text analysis of open source software defect reports," *Empirical Software Engineering*, 2012.
- [12] "SAS Text Miner," SAS, [Online]. Available: <http://support.sas.com/software/products/txtminer/>. [Accessed 16 2015].
- [13] Emanuel Giger, Martin Pinzger, Harald Gall, "Predicting the fix time of bugs," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010.
- [14] Hui Zeng, David Rine, "Estimation of software defects fix effort using neural networks," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International (Volume:2)*, 2004.
- [15] Lucas D. Panjer, "Predicting Eclipse Bug Lifetimes," in *Mining Software Repositories, 2007. ICSE Workshops MSR '07.*, Minneapolis, MN, 2007.
- [16] W. Abdelmoez; Mohamed Kholief; Fayrouz M. Elsalmy, "Bug Fix-Time Prediction Model Using Naïve Bayes Classifier," in *Computer Theory and Applications (ICCTA), 2012 22nd International Conference*, Alexandria, 2012.
- [17] Lionel Marks; Ying Zou; Ahmed E. Hassan, "Studying the Fix-Time for Bugs in Large Open Source Projects," in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, 2011.
- [18] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows," in *Software Engineering (ICSE), 2012 34th International Conference*, Zurich, 2010.

- [19] Pamela Bhattacharya, Iulian Neamtiu, "Bug-fix Time Prediction Models: Can We Do Better?," in *Software Engineering (ICSE), 2013 35th International Conference*, San Francisco, CA, 2011.
- [20] Prasanth Anbalagan; Mladen Vouk, "On Predicting the Time taken to Correct Bug Reports in Open Source Projects," in *Proceedings of IEEE International Conference on Software Maintenance (ICSM 2009)*, 2009.
- [21] Ahmed Lamkanfi, Serge Demeyer, "Filtering Bug Reports for Fix-Time Analysis," in *Software Maintenance and Reengineering (CSMR), 2012, IEEE*, Szeged, 2012.
- [22] W. AbdelMoez, Mohamed Kholief, Fayrouz M. Elsalmy, "Improving Bug Fix-Time Prediction Model by Filtering Out Outliers," in *Technological Advances in Electrical, Electronics and Computer Engineering (TAECE), 2013, International Conference, IEEE*, Konya, 2013.
- [23] Kurt Hornik; Johannes Rauch; Christian Buchta; Ingo Feinerer, "Package 'textcat': N-Gram Based Text Categorization," 2 6 2015. [Online]. Available: <http://cran.r-project.org/web/packages/textcat/textcat.pdf>. [Accessed 7 6 2015].
- [24] "Toggl - Free time tracking software & app," Toggl, [Online]. Available: <https://www.toggl.com/>. [Accessed 7 6 2015].
- [25] "Class Similarity," Apache Software Foundation, 2000-2010. [Online]. Available: <https://lucene.apache.org/>. [Accessed 1 5 2015].
- [26] "Cosine Similarity Calculator," Applied Software Design, 13 4 2012. [Online]. Available: <http://www.appliedsoftwaredesign.com/archives/cosine-similarity-calculator/>. [Accessed 3 8 2015].
- [27] J. R. Quinlan, C4.5: programs for machine learning, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.
- [28] Leo Breiman; Adele Cutler, "Random forests - classification description," Department of Statistics, University of California, [Online]. Available:

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed 6 6 2015].

- [29] M. J. Norušis, "Chapter 4: Ordinal Regression," in *PASW Statistics 18.0 Advanced Statistical Procedures Companion*, 2010, p. 648.
- [30] M. Porter, "The Porter Stemming Algorithm," 2006. [Online]. Available: <http://tartarus.org/martin/PorterStemmer/>. [Accessed 4 4 2015].
- [31] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference*, Beijing, 2012.
- [32] "What is LSA?," University of Colorado Boulder, [Online]. Available: <http://lsa.colorado.edu/whatis.html>. [Accessed 25 5 2015].
- [33] Lailil Muflikhah; Baharum Baharudin, "Document Clustering using Concept Space and Cosine Similarity Measurement," in *Computer Technology and Development, 2009. ICCTD '09. International Conference*, Kota Kinabalu, 2009.
- [34] A. Utsumi, "Evaluating the Performance of Nonnegative Matrix Factorization for Constructing Semantic Spaces: Comparison to Latent Semantic Analysis," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference*, Istanbul, 2010.
- [35] Rui Máximo Esteves; Chunming Rong, "Using Mahout for clustering Wikipedia's latest articles," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference*, Athens, 2011.
- [36] "Lucene Apache Org," Apache, 2015. [Online]. Available: <http://lucene.apache.org/>. [Accessed 25 3 2015].
- [37] O. Gross, "Finding Non-Trivially Similar Documents from a Large Document Corpus," University of Tartu, Tartu, 2011.

- [38] Strehl, Alexander; Ghosh, Joydeep; Mooney, Raymond;, "Impact of similarity measures on web-page clustering," in *In Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, 2000.
- [39] P. J. Rousseeuw, "Silhouettes: a raphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. 2, pp. 53-65, 1987.
- [40] M. Jørgensen, "What We Do and Don't Know about Software Development Effort Estimation," *Software, IEEE*, vol. 31, no. 2, pp. 37 - 40, 2014.

Appendix

I. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Myroslava Stavnycha** (date of birth: 10.07.1992),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

- 1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright

of my thesis

Issue Report Resolution Time Prediction,

supervised by Dietmar Pfahl,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **04.08.2015**