Thesis no: MSCS-2016-10

Performance evaluation based on data from code reviews

Andrej Sekáč

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science of Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Andrej Sekáč
E-mail: ansa14@student.bth.se

External advisor:
Dr Lars-Ola Damm
Ericsson AB

University advisor:
Dr Julia Sidorova
Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Context. Modern code review tools such as Gerrit have made available great amounts of code review data from different open source projects as well as other commercial projects. Code reviews are used to keep the quality of produced source code under control but the stored data could also be used for evaluation of the software development process.
Objectives. This thesis uses machine learning methods for an approximation of review expert's performance evaluation function. Due to limitations in the size of labelled data sample, this work uses semi-supervised machine learning methods and measure their influence on the performance. In this research we propose features and also analyse their relevance to development performance evaluation.
Methods. This thesis uses Radial Basis Function networks as the regression algorithm for the performance evaluation approximation and Metric Based Regularisation as the semi-supervised learning method. For the analysis of feature set and goodness of fit we use statistical tools with manual analysis.
Results. The semi-supervised learning method achieved a similar accuracy to supervised versions of algorithm. The feature analysis showed that there is a significant negative correlation between the performance evaluation and three other features. A manual verification of learned models on unlabelled data achieved 73.68% accuracy.
Conclusions. We have not managed to prove that the used semi-supervised learning method would perform better than supervised learning methods. The analysis of the feature set suggests that the number of reviewers, the ratio of comments to the change size and the amount of code lines modified in later parts of development are relevant to performance evaluation task with high probability. The achieved accuracy of models close to 75% leads us to believe that, considering the limited size of labelled data set, our work provides a solid base for further improvements in the performance evaluation approximation.

Keywords: software verification, semi-supervised learning, regression analysis, development performance evaluation

Acknowledgements

I would like to thank Dr Julia Sidorova for valuable input and feedback that helped with finishing this work. Furthermore, I would like to thank Dr Lars-Ola Damm for a great help and inspiration with defining the focus of this thesis and especially for opening the environment in the company thus making this research possible. Also I would like to thank my relatives and friends for support during this work.

Finally, I would like to express my great gratitude to the nature that surrounds us and especially the nature of Karlskrona that provided support and inspiration to me.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

With the introduction of modern code review tools, such as Gerrit [2], the code reviews are becoming an integral part of development in many projects including large Open Source projects such as Qt [5]. Modern code review tools compared to traditional code reviews as described by Fagan in 1976 [16] are more lightweight and have an advantage of storing data about code reviews performed in a easily accessible way. This newly accessible code review data opens a great number of research possibilities. This thesis aims to utilise the code review data available to evaluate the underlying development process. The work was done in cooperation with the large telecommunication company Ericsson, which enabled the access to its review expertise and code review data.

1.1 Problem

The research done in the field of code review data analysis shows that it is possible to study the influence of reviewers' project role on the review outcomes [19], or defect proneness of software depending on the review coverage [26]. The ambition of this thesis is to study the possibility of using the available data about code reviews to evaluate how well did the code contributors perform during the development process.

To grasp the performance evaluation in an easy to express way, it is represented by an evaluation metric, which grades the performance on a numerical scale. The task is then to automatically predict an accurate performance grade for a given code review data. More specifically the aim is to simulate a performance evaluation of an expert reviewer. Thus, the goal of this thesis is to design a machine learning solution for approximation of the reviewers' performance evaluation function using the code review data. That is a well known problem of regression for which a number of different solutions exists. From the family of machine learning techniques it can be achieved with a multi-layer perceptron, a radial-basis function network or other [17, 31, 29, 12].

The regression task requires labelled data, a sample of known values of the unknown function, which is for the purpose of performance evaluation not available by default. This problem has been mitigated by a manual labelling process performed by Ericsson's review experts. However, as the time of review experts is extremely expensive, the available labelled sample is limited in size. To further mitigate the insufficiency of the sample size, a semi-supervised machine learning approach has been used. The previous research proves that it is possible to achieve a good performance using the semi-supervised learning even on smaller labelled dataset under the condition that enough of the unlabelled data is available [38, 13, 15]. The database contains several years of development and the unlabelled data is available in great amounts.

Finally, as a part of the machine learning solution design a set of features has been designed. The work aims to use the labelled data to analyse the influence and explanatory value of each feature on the evaluation. This, done by a manual analysis with the use of statistical tools and by creating a decision tree, provides complementary knowledge to the previous research.

1.2 Gerrit Code Reviews

Gerrit is a modern code review tool based on GIT version control system [2, 3]. When a newly developed code is committed using GIT, a Gerrit change instance is created in Gerrit. This Gerrit change is then waiting in Gerrit for approval. Reviewers use a web interface to access Gerrit and review the submitted code. Gerrit allows for leaving comments on selected parts of the code and leaving approval ratings. Ratings can be -2, -1, +1, +2. The change committer can respond to received feedback using the Gerrit, and he or she can submit a new code as a patchset to the existing change. The change waits in Gerrit until it is approved with the +2 approval rating or until it is abandoned. If a change receives the +2 approval rating it is approved and merged into the upstream branch.

Therefore the general workflow starts with a change being created by submitting a newly committed code. Then the author of the change or contributor awaits feedback from reviewers. If the feedback is negative and the change receives negative ratings the author sends a new patchset to address the problem identified during the review and again waits for feedback. If the feedback is positive and change receives a +2 rating, it is merged. Any developer can submit a new patchset to a change even if he or she is not the owner of that change. Henceforth the whole review process of a single change, including all patchsets' reviews, is considered to be a single research unit and called a review.

The data available comes from one of Ericsson's large scale projects with a long history. The Gerrit has been used within this project for several years and currently its database contains a long track record of several thousands of changes. The development process in this project has some specific rules. Every single change has to be reviewed and approved before merging, which is also encouraged in research by Tanaka et al. [34]. The project includes several development teams and each team has its leader who is responsible for the work produced. A new change from a team should be first reviewed by fellow team members and then approved by the team leader. Usually only the team leader has the right to put the +2 approval rating during a review. Furthermore, the project has one team of review experts, which helps with reviewing critical components of the product as well as with problematic changes. If a developer or a team leader is unsure with the performed review, they can ask the expert team for a review help. The reviewers in the expert team are experienced in reviewing and have a high-level knowledge of the project.

1.3 Regression

The regression task is a class of problems with the following components. A given unknown function f, which maps an input from input space X to an output space Y, where

$$f : X \to Y, \{Y \subseteq \mathbb{R}\} \tag{1.1}$$
$$f(x) = y$$
$$h(x) \approx y,$$

and the goal is to find a model h such that it would approximate the origin function f as accurately as possible [33]. The possible quality of the model h is strictly dependent on the representation of X. That makes it important to define the features representing the input as fitting to the problem of performance evaluation as possible.

1.4 Supervised Machine Learning

A standard supervised machine learning task is to learn a model using training data, in which input data points with desired output values are available. The final task then depends on the representation of the output values. If the output values are real numbers, the task could be the regression where the goal is to fit a function to the known labelled data points. If the output values are values from a finite set of labels, the task could be classification, where the goal is to predict a correct label for a input data point. The supervised learning relies on having the expected output values for the training dataset available [28, 35].

1.5 Semi-supervised Learning

The semi-supervised learning is a machine learning approach that combines elements of supervised learning and unsupervised learning [13]. It has two perspectives. The first is the perspective of unsupervised learning with additional constraints. This setting considers having unlabelled data with some additional information, such as labels or other. Constrained clustering algorithms introduced by Abu-Mostafa are an example application [9]. The second perspective is the supervised learning with additional information on the distribution of the input data. This is the setting where the available data can be divided into a significantly smaller set with known labels and the remaining set of unlabelled data. It is considered as the standard semi-supervised learning setting and the studied problem fulfils its description. The research suggests that if certain conditions are met, the semi-supervised learning can lead to a greater accuracy for algorithms when compared to their supervised or unsupervised variations [13].

1.6 Aims and Objectives

The aim of this work is

1. to design and implement a machine learning technique, which would use data from the code reviews to approximate experts' quantitative evaluation of software developers' coding performance,

2. to investigate influence of using semi-supervised machine learning paradigm on the code review data domain in comparison to supervised learning,

3. analyse the influence of selected features on the model.

The available data is from a real world working environment at Ericsson.

To achieve aims formulated above, the problem is split into several subtasks, each representing an objective necessary to achieve.

- Define a set of features for input space representation.
- Select a representative sample and obtain labels for it.
- Choose a semi-supervised learning approach as a wrapper for a suitable machine learning algorithm.
- Implement the chosen algorithm in semi-supervised framework and test it together with its supervised variation.
- Analyse the results of test and compare the semi-supervised and supervised variations.
- Analyse the influence of selected features.

1.7 Research Questions

RQ1 Can the semi-supervised machine learning approaches be beneficial when used for approximation of an expert's performance evaluation?

The first research questions aims to find whether it is possible and useful to use the semi-supervised machine learning in the studied task. Utilising the great amount of unlabelled data in the learning process could bring valuable improvements to accuracy that otherwise could be limited by the small size of the labelled data.

RQ2 Which features are relevant for the performance evaluation task?

The second research question asks, which of the data metrics available in the Gerrit database have influence on the performance evaluation predictions. Answering this question can not only play role in the function approximation task, but also can give a base for future code review data analysis or other research.

RQ3 Is it possible to approximate an expert's performance evaluation by using machine learning methods on the data from code reviews?

The last research question is trying to give a verdict on whether the code review data has a sufficient explanatory power to accurately predict the software development performance evaluation. If the learned models prove to be accurate enough, performance evaluation values could be predicted for a great amount of Gerrit changes, which would enable further analysis of the software development performance.

1.8 Thesis Outline

The remainder of the thesis is organised as follows.

Chapter 2 discusses the related work done in the field of code review data analysis and semi-supervised regression analysis.

Chapter 3 describes in detail the methods used to perform this research. It explains the data gathering process, the feature selection process, the details of machine learning algorithms, experimental settings and the feature analysis.

Chapter 4 presents the results of the performed experiment and analysis.

Chapter 5 analyses the results and discusses the implications of results and possible outcomes of the research.

Chapter 6 draws conclusions and discusses possible interesting topics for the future work.

Chapter 2

Related Work

The following chapter discusses the existing research related to the studied problem. Due to the lack of work related to both regression based on semi-supervised machine learning and the analysis of data from code reviews, the chapter is split into two sections. The first of them describes related work from the code reviews analysis perspective. And the second gives an overview of work related to the semi-supervised regression task.

2.1 Code reviews analysis

McIntosh et al. studied the impact of code review coverage and participation on the incidence of post-release defects in software [26]. For this purpose they gathered a number of different metrics describing the source code characteristics as well as the review coverage and review participation. Then they built Multiple Linear Regression (MLR) models for predicting the number of post-release defects of a component. Afterwards they observed explanatory values of each metric and drew the conclusion that modules with a lower proportion of reviewed changes and with lower review participation tend to have more post-release defects, even though a good review coverage or participation does not prevent post-release defects completely.

A similar study performed by Morales et al. [30] studies the impact of code review coverage and participation on the incidence of design anti-patterns in modules. An anti-pattern is a software development technique, which has been proven to lead to complications in further development [22]. Anti-patterns are considered as the typical bad-practice software development caveats. Authors in this study gathered a list of metrics describing source code characteristics and code review characteristics. To study the influence and explanatory value of designed metrics on the anti-pattern incidence they built Ordinary Least Squares (OLS) regression models. After examination of the built models they concluded that there exists sufficient evidence that in at least two out of four studied projects a poor code review coverage and a lack of review participation lead to a grow in anti-pattern occurrence.

The work by Meneely et al. [27] studies a specific set of code review metrics and their influence on detecting security vulnerabilities. The data analysed comes from open source project Chromium and its Rietveld code review tool [1, 7]. The aim of their research is to verify from a security perspective, whether the standard open source development paradigm, i.e., many developers working on one piece of code ensure low bug incidence. The results show that greater number of reviewers does not guarantee a less vulnerable code if the reviewers are not experienced in security vulnerabilities detection.

Another work by Liang et al. [25] analysed the code review database of Chromium project and found statistically significant links between the number of reviewers and the number of patchsets. Changes with greater number of reviewers tend to have more patchsets. Also the authors found some evidence suggesting that smaller changes are more likely to be chosen for review earlier than bigger ones.

A research done by Kula et al. [24] displays the use of Gerrit database from Android Open Source Project to profile individual developers and reviewers. In their research they managed to design several metrics and use them to evaluate an individual's role and presence in the open source project.

2.2 Semi-supervised Regression

Zhou et al. in their paper from 2005 proposed a new semi-supervised learning approach of regression analysis [37]. In their study, they present a learning algorithm based on co-training and k-nearest neighbours (kNN) regression models. The algorithm is based on the co-training idea, where two independent regression models train each other. When updating the model of the first one, the second regression model labels an example from unlabelled data, then the first regression model uses its extended labelled dataset for the training and vice versa. The proposed algorithm extends the co-training paradigm by a novel selection of unlabelled data examples. When deciding, which unlabelled example to choose for the next iteration of training, the algorithm uses the kNN calculation for each unlabelled example to find the most confidently labelled example. The experimental data presented proves that the presented *COREG* algorithm can outperform other standard semi-supervised algorithms and provide increased accuracy compared to supervised models on almost all presented datasets. It is also noteworthy that *COREG*, unlike standard co-training algorithms, is not constrained by the requirement of sufficient and redundant views, which makes it applicable on a wider spectre of problems.

Recently Kang et al. proposed a semi-supervised regression algorithm based on the self-training semi-supervised learning approach and support vector regression algorithms [21]. The algorithm first estimates the distribution of label for each unlabelled data point. Then it generates new training data by using the estimated distributions scaled to the uncertainty of the estimation i.e. the variance of the distribution. Afterwards, a final smaller training sample is selected from the generated data to lower the complexity of learning caused by a large size of training data. Finally, a standard support vector regression algorithm is used to train the model. The experimental results show that this algorithm can outperform some other semi-supervised regression algorithms, such as $COREG$ [37]. Furthermore the algorithm avoids using iterative learning as opposed to $COREG$, which lowers the learning process time significantly.

Chapter 3

Method

This chapter describes the details of all methods used to perform the research.

3.1 Data gathering

The data used in the study comes from the Gerrit instance of one of large Ericsson's projects. The data from Gerrit is stored in an internal Gerrit database and is accessible via REST API documented online. For an easier and more straightforward access, necessary data has been downloaded to a local database. For this purpose a MySQL database has been used [4], into which the Gerrit database scheme has been mapped. The downloading process was managed by a Java application implemented on a base of the same purpose Java application from the study by Grisins and Sekac [19]. The implementation uses an existing Java REST API client to interact with Gerrit and then stores the received data into a MySQL database. To avoid possible misuse and influence on the research, the identities of developers have been removed from the data during the loading process.

3.2 Feature design and selection

To select input vector features, we have gathered a preliminary list of data metrics available from the code review data, which could possibly have a explanatory value for the performance evaluation task. These were based on the knowledge gained from related research of code review data [26] as well as on the researcher's own knowledge. Then a interview with Ericsson's review experts was appointed to further discuss the feature selection. This lead to a selection of 10 different data metrics as the features for the machine learning task. Table 3.1 describes the selected features and discusses the motivation of selection for each of them.

To further validate the feature selection we used a Spearman's rank correlation test [36]. We have selected a sample of last 1000 changes from Gerrit ordered by the creation date and generated feature vectors for them. Then we have ran the Spearman's rank correlation test using the `cor` function from R. As a border value

Table 3.1: Initial feature selection with description

Feature	Description	Rationale
Contributors	Number of contributors.	A greater number of contributors can be a sign of complications in the development process or increased complexity of the task.
Reviewers	Number of reviewers.	A greater number of reviewers can be a sign of complications in the development process or increased complexity of the task.
Size	Number of lines of code.	A large size is linked to more defect-prone software [23]. An unnecessarily large size can be caused by a poor development.
Comment ratio	Ratio of comments to size of change.	Comments in a review are a solid evidence of performed review. A very low ratio can be a sign of low review coverage.
Reply ratio	Ratio of comments with response to all comments.	Comments with responses are the evidence for communication between reviewers and authors. A lack of communication can be a sign of a worse development process.
Response time	Average response time to comments.	A longer response time by the change author can reflect problems in the development.
Self-approved	Value set to true if change owner is also approver.	If a change is approved by its author the review might not have sufficiently covered the modified code.
Files	Number of files modified.	A greater amount of modified files can be descriptive of a complex problem or a suboptimal work division.
Churn	Sum of all modified lines the first patchset.	A larger amount of necessary changes after the first patchset can be linked to a worse development process.
Waste	Sum of all removed lines the first patchset.	The code waste can partly express the amount of code that was produced and discarded during a change. A lot of code produced and then removed in a change can be a sign of inefficient development.

Table 3.2: Spearman's rank correlation coefficients

	Con	Rev	Siz	Com	Rep	Res	Sel	Fil	Was	Chu
Con	1									
Rev	0.03	1								
Siz	0.03	0.11	1							
Com	-0.00	0.46	-0.04	1						
Rep	0.02	0.62	0.16	0.43	1					
Res	-0.05	-0.61	-0.17	-0.42	**-0.92**	1				
Sel	0.01	-0.28	0.02	-0.14	-0.07	0.05	1			
Fil	0.00	0.00	0.38	-0.01	0.02	-0.02	0.07	1		
Was	0.01	0.32	0.55	0.14	0.22	-0.22	-0.05	0.09	1	
Chu	0.02	0.35	0.48	0.14	0.24	-0.23	-0.06	0.08	**0.95**	1

we set `p=0.7`. Table 3.2 shows the Spearman's coefficients. The coefficient value is greater than 0.7 in two cases. The first case is `reply ratio` and `response time` features. The correlation between them was statistically significant and therefore we have decided to keep only `reply ratio` feature as it is more broad. The second case is `churn` and `waste` features. Their correlation was statistically significant as well and we have decided to discard `waste` from the final feature list as `churn` is a more general metric and catches also addition changes.

3.3 Data preprocessing

Due to the fact that several of the features were varying on great scales and included a lot of outliers, we have decided to use a logarithmic transformation some of the features [35]. We have used the logarithmic transformation on all features except for the `comment ratio` and `reply ratio` features, which are ratios of values between 0 and 1.

3.4 Data labelling

Initially, the data available was all unlabelled with respect to the performance evaluation. To get labels for data samples, a labelling metric had to be defined and a small sample of examples had to be chosen. For the labelling metric definition an interview with Ericsson's experts was held. The discussions lead to a conclusion that a more fine grained performance evaluation metric would significantly increase the labelling time. Due to the fact that the available time for the data labelling was strictly limited, we agreed on a rather coarse and simple metric. The development changes were graded on a five point scale 1, 2, 3, 4, and 5, which was afterwards rescaled to a scale from -1 to 1 where

- -1 means a rather poor development performance,
- -0.5 means a below average quality of development performance,
- 0 means a development performance of medium quality,
- 0.5 means a good development performance, and
- 1 means a very good development performance.

This metric, although simple, allows for a faster decision making during the evaluation labelling process leading to a larger labelled sample. As the time limitations for labelling turned out to be very strict, it was impossible to adopt a more fine grained metric, which could be more interesting from the performance evaluation perspective.

After the evaluation metric was selected a representative sample was selected for the labelling process. Two Ericsson's review experts have been assigned to perform the labelling process, each of them capable of labelling 25 examples. The final labelled sample could be at most 50 change examples large. For the sample selection we have set a number of constraints. The samples have been selected only from the first 200 changes ordered by the creation date, which represents approximately three months. This constraint takes in consideration the following fact: the older a change is, the higher is the bias in the evaluation process, as the reviewers might not remember all the details of each particular task. Furthermore, to achieve a more precise evaluation, each selected example had to be previously reviewed by one of the assigned review experts. For this reason the list of first 200 changes has been further filtered for changes with at least one of the review experts as reviewers and then divided into two groups for each review expert. Then from each group 25 examples were manually chosen in an attempt to keep the variance of data features as high as possible. The sample was chosen in the beginning of February 2016. Then the selected sample was sent to the review experts for labelling.

As the unlabelled data set for the machine learning task we used the first 1000 changes ordered by creation date excluding the changes included in the sample chosen for labelling.

3.5 Regularisation in Semi-supervised learning

As the semi-supervised learning method for solving the regression machine learning problem we have chosen the metric-based regularisation as proposed by Schurmans and Southey in [32]. The motivation for the choice was that the presented experimental results show that the method performs well, improves the regression accuracy and also represents a novel approach to unlabelled data utilisation.

This method is based on creating a measure in the metric space of hypotheses that can be used to compare individual hypotheses' distance from an approximation of the optimal hypotheses. That can be used for regularisation e.g. finding good fitting regularisation parameters. The authors tested the method for regression tasks on multiple datasets and proved that it in a great majority of problems brings improvement in the precision of learned regression model.

For measurement of distance between two particular hypotheses f and g from the hypotheses space we use

$$d(f,g)\hat{=}(err(f(x),g(x)))^{\frac{1}{2}}, \tag{3.1}$$

where err is an error function. As the error function we use Mean Square Error function (MSE)

$$err(Y',Y) = \frac{1}{n}\sum_{i=1}^{n}(y'_i - y_i)^2, \tag{3.2}$$

which gives final distance equation

$$d(f,g)\hat{=}(\frac{1}{n}\sum_{i=1}^{n}(f(x_i) - g(x_i))^2)^{\frac{1}{2}}, \tag{3.3}$$

where x_i is an example from the set of labelled data. For the function measuring distance between two individual hypotheses on the set of unlabelled data is used notation $\hat{d}(f,g)$ in which x_i are drawn from the set of unlabelled data.

The regularisation method then tries to use available unlabelled data for better estimation of regularisation parameters. This is done by taking in consideration not only the estimated distance of current hypothesis to the optimal hypothesis, but modifying it further by the difference in behaviour of the current hypothesis on labelled and unlabelled data. This difference is expressed by measuring the distance of current hypothesis to a certain origin function ϕ in labelled data points and unlabelled data points. An origin function can be for this purpose almost any trivial function e.g. $\phi = \bar{y}$ a constant mean function.

The mechanism utilises the unlabelled data by penalising hypotheses, which behave significantly different on labelled data and unlabelled data. To achieve that, the following term is used in an optimisation task.

$$d(h,P_y) \times max(\frac{d(h,\phi)}{\hat{d}(h,\phi)}, \frac{\hat{d}(h,\phi)}{d(h,\phi)}), \tag{3.4}$$

where h is the current hypothesis and P_y is the estimated optimal hypothesis.

3.6 Radial Basis Function Network

We have chosen the Radial Basis Function (RBF) network as the regression machine learning algorithm. This was because RBF networks often require regularisation, which makes them suitable for implementation with the metric-based regularisation method described in the previous section. Furthermore RBF networks can be fitted by solving linear equations, which can make the fitting procedure significantly less time consuming [31, 32].

RBF networks were introduced by Broomhead in [11] and can used for function approximation tasks. For a given set of k prototypes used as centres $c_1, ..., c_k$ a RBF function h fulfils

$$h(x) = \sum_{i=1}^{k} w_i g(\frac{||x - c_i||}{\sigma}), \tag{3.5}$$

where w_i is a centre weight from a weight vector w of length k, g is a basis function with a width parameter σ, and $||x - c_i||$ is a Euclidean distance between the centre c_i and example x. In this work the basis function is a Gaussian function $g(x) = e^{\frac{x^2}{\sigma^2}}$.

RBF networks are then fitted to a problem by using labelled sample X as centres and solving for w

$$h(X) \times w = y, \tag{3.6}$$

where y is the label vector for X. However, such solutions usually lead to overfitting and therefore the RBF network fitting requires regularisation [14]. One of the basic regularisation techniques is the ridge regularisation. It adds a ridge penalty to the weight vector in a form of additional parameter λ. The goal is then to minimise the following expression

$$\sum_{i=1}^{l}(h(x_i) - y_i) + \lambda \sum_{i=1}^{l} w_i^2, \tag{3.7}$$

where l is the number of labelled samples.

The supervised learning approach relies on fitting the RBF networks using a particular combination of values for parameters σ and λ. That can be a non-trivial task and require deeper knowledge about the particular dataset. The semi-supervised regularisation framework can be used to automate the selection of regularisation parameters in a data-oriented way [32]. To apply the method described in the previous section, we have to run an optimiser over the parameter space of σ and λ and solving for a w vector that minimises (3.7). Then for each set of values σ, λ and w we provide the value of (3.4) as the objective to minimise by the optimiser.

3.7 Experiment

The experiment to address RQ1 consists of comparison between the supervised and semi-supervised variations of RBF. The supervised variation of RBF uses a standard optimiser to minimise (3.7) for a given values of σ, λ. For an extensive comparison the supervised variation is tested for 35 different pairs of regularisation parameters. Each test is run 100 times by executing a 10-fold cross validation with a random split of the labelled data sample to 9/10 used as training set and 1/10 as testing set. The semi-supervised version uses the metric-based regularisation as described in the previous section and same test setting as supervised variations except for using also the unlabelled data set. The final average error values from each test are then compared to determine the lowest value.

For evaluation of goodness of fit for the final model we have calculated the R^2 and adjusted-R^2 metrics [18, 20]. Then we assessed the acquired values to estimate the accuracy of model.

Furthermore, the labels of data have been modified to a string value so that classification algorithms could be used. Labels -1, -0.5, 0, 0.5, 1 have been mapped to `worst, bad, medium, good, best` respectively. Afterwards a decision tree has been built using the J48 algorithm implemented in Weka Explorer environment [8]. A decision tree model is represented by decision nodes organised in a binary tree where the leaves are the final labels. Each node has one feature and a threshold value, which it uses to choose the next node. The main advantage of the decision tree is that the rules used for decisions are easily understandable and can be used to see, which of the features are used in the decision making process. This can help to analyse the relation of particular features to the grade value.

3.8 Analysis

To answer RQ2 the labelled data has been drawn to a dependency chart between the label and each feature, resulting in 8 scatter plots. These has been manually analysed and then statistical tools have been used to verify observed behaviour. Then a decision tree has been generated and analysed to further verify the findings about the feature relevance.

A verification of a label is significantly less time consuming as compared to full labelling process. Thus, to verify the functionality of the learned model we have labelled a random sample from the unlabelled dataset and asked for verification by Ericsson's review experts. This gives further information for answering RQ3.

Chapter 4

Results

This chapter presents results of performed experiment and analysis. According to the description in Chapter 3 we have gathered data, created feature vectors, selected sample for labelling and requested labels. However, due to time constraints the received labelled sample contained only 40 examples.

4.1 Experiment

We have performed the experiment by running the 10-fold cross validation learning 100 times for each regularisation setting and gathered the results in the Table 4.1. ADA describes the semi-supervised learning algorithm and REG standard supervised version. The first row shows the MSE and standard deviation of ADA and the best performing REG. It can be seen that the MSE of ADA is 0.2263 while the best result of REG is 0.2251 as in the case of $\sigma = 9$ and $\lambda = 0.01$. The difference between the best MSE of REG and ADA is not significant.

To evaluate the goodness of fit of produced models we have calculated R^2 and adjusted-R^2 values for the model with the lowest MSE. The value of R^2 was 0.5393, which means that model explains 53.93% of the data variance. The value of adjusted-R^2 was 0.4385, which adjusts the R^2 value with regard to number of explanatory variables used. The significantly lower adjusted-R^2 value suggests that some of the features do not contribute positively to the regression accuracy.

Listing 4.1: Decision tree structure

```
reviewers <= 1.94591
|    contributors <= 0.693147
|    |    churn <= 6.111467: best (20.0/5.0)
|    |    churn > 6.111467
|    |    |    comment_ratio <= 0.203349: good (9.0/2.0)
|    |    |    comment_ratio > 0.203349: medium (5.0/2.0)
|    contributors > 0.693147: medium (2.0)
reviewers > 1.94591
|    size <= 5.463832: medium (2.0)
|    size > 5.463832: worst (2.0)
```

Table 4.1: Mean Squared Error values for different regularisation settings

ADA 0.2263 ±0.0126			**REG** 0.2251 ±0.0097		
REG	$\lambda = 0.01$	0.1	0.25	0.5	1.0
$\sigma = 3$	0.3237	0.2732	0.2627	0.2572	0.2520
4	0.2852	0.2524	0.2437	0.2385	0.2355
5	0.2630	0.2383	0.2311	0.2299	0.2316
6	0.2475	0.2285	0.2283	0.2298	0.2301
7	0.2349	0.2276	0.2280	0.2303	0.2317
8	0.2282	0.2269	0.2273	0.2283	0.2333
9	**0.2251**	0.2285	0.2280	0.2294	0.2362

Finally, to provide additional information for feature analysis we have modified the evaluation metric to suit a classification problem. Then we have used Weka to generate a decision tree using J48 algorithm with 10-fold cross-validation. Listing 4.1 provides the final structure of learned and pruned decision tree as received from Weka Explorer interface. The output structure shows in parentheses how many examples end in that node and how many examples are misclassified if any. The decision tree shows that on the labelled data sample it classifies incorrectly 9 out of 40 examples. The classification accuracy is 77.5%.

We can see that the decision tree uses five out of eight features, more specifically `reviewers, contributors, size, comment ratio`, and `churn`. According to the decision tree it is important for the better graded examples of `best, good` to have a lower number of reviewers, contributors and lower `churn` or `comment ratio` values. On the other hand the worst graded examples are classified by having a higher number of reviewers and a greater size.

4.2 Statistical analysis

For the statistical analysis we have split the labelled data into two groups. One group with the grade higher than 0.25, which are considered to be a sign of good or very good performance, and the other group with the grade lower or equal to 0.25, which are considered to be representative of average or poor performance.

First we have generated scatter plots of dependencies between the grade and each other feature. Plots for `reviewers, comment ratio` and `churn` seemed to show differences in data between the two observed groups as it can be seen on Figure 4.1. Plots for other features have been excluded for brevity.

To investigate the differences between observed groups we have used `t.test`

Table 4.2: Mean values of individual features and their significance

Feature	grade > 0.25	grade <= 0.25	p-value
Contributors	0.6931	0.7607	0.1661
Reviewers	1.2848	1.7605	**0.0106**
Size	5.5119	5.5695	0.9139
Comment ratio	0.1283	0.5587	**0.0016**
Reply ratio	0.7277	0.8488	0.1405
Self-approved	0.4642	0.2500	0.1989
Files	2.3804	2.2365	0.6682
Churn	5.1792	6.8669	**0.0022**

function from `R` on every feature [6]. Table 4.2 shows results of performed tests. Differences in feature means between observed groups were statistically significant, $p - value < 0.05$, in the same three cases as we have observed in plots.

However, means of `reviewers` and `churn` are rescaled by using the logarithmic transformation as described in Chapter 3. For later analysis of the results we have performed also the inverse transformation. For the `reviewers` feature, means are approximately 3.6 and 5.8 for the > 0.25 group and $<= 0.25$ group respectively. For the `churn` feature, means are approximately 178 and 960 for the > 0.25 group and $<= 0.25$ group respectively.

4.3 Verification procedure

For review experts' verification we have selected a sample of 40 changes reviewed by the review experts and labelled them using the best fitting regression model from the experiment section. These newly labelled samples have been sent for verification to the review experts. Those had reviewed the labels and verified the correctness of their prediction. Table 4.3 shows results of the verification process. Two of the selected examples could not be verified due to various reasons. From the remaining sample counting 38 examples, 28 labels have been positively verified while 10 labels have not been precise. That results in 73.68% of examples labelled successfully by the learned model.

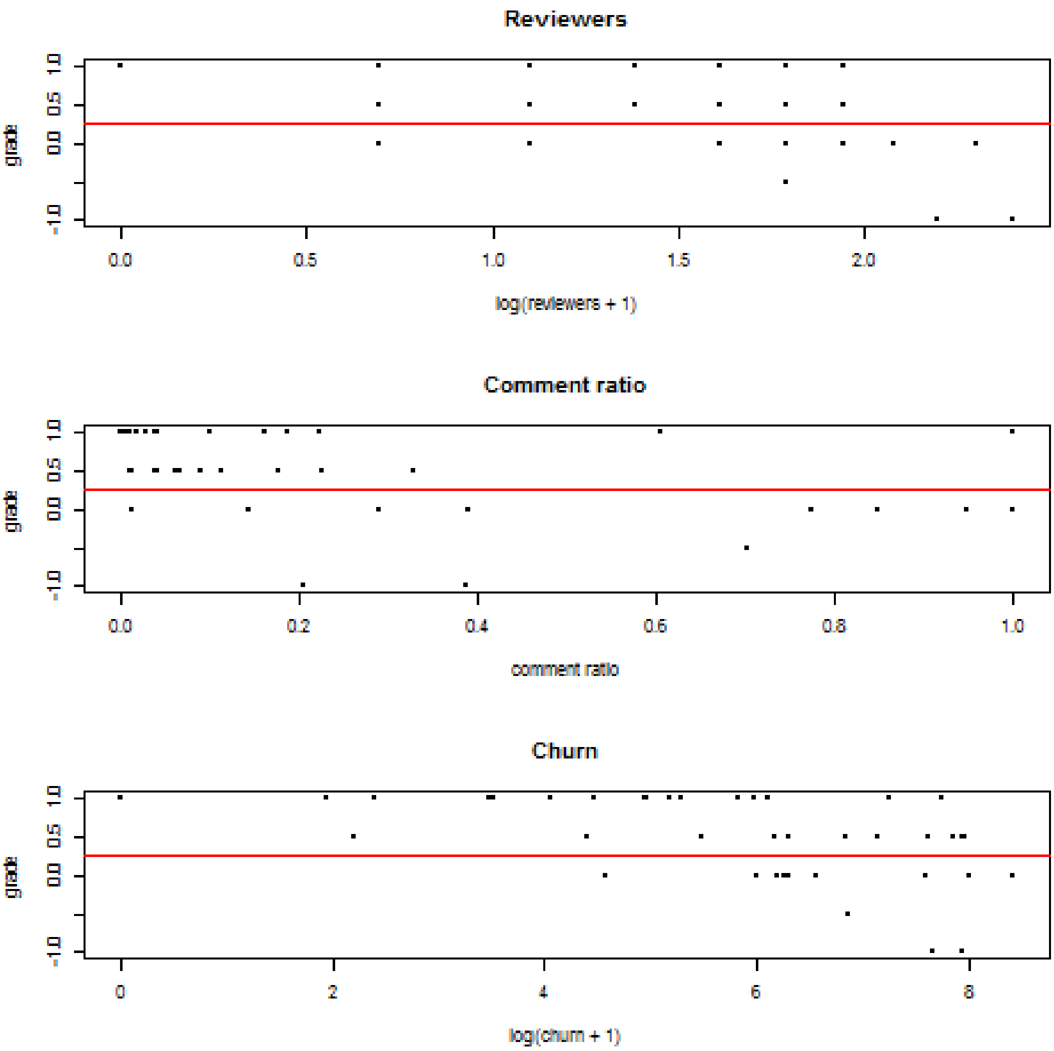

Figure 4.1: Relations of individual features and the grade values

Table 4.3: Manual verification results

Sample size	Verified examples	Correctly predicted	Accuracy of labelling
40	38	28	73.68%

Chapter 5

Analysis and Discussion

This chapter presents the analysis of obtained results and discusses possible conclusions. At the end the threats to validity are discussed. The chapter is split into parts, each discussing results from a perspective of one of the research questions.

5.1 Experiment (RQ1)

The results obtained from the performed experiment show that the used semi-supervised method did not bring improvements to the performance of regression model. However, the difference between MSE of the best supervised model and the semi-supervised model is rather small, and therefore it seems that the performance did not degrade with the use of semi-supervised method. The improvements from using the semi-supervised method could be limited by the data characteristics and also by the rather small size of labelled data sample.

Thus considering the available data, selected features, selected semi-supervised learning method and selected regression analysis algorithm, the obtained results do not support the claim that semi-supervised learning method would be beneficial for the approximation of the review experts' performance evaluation function in terms of accuracy. A greater size of labelled sample as well as a different semi-supervised method could bring different outcomes.

5.2 Feature analysis (RQ2)

To estimate, which features are relevant for the performance evaluation regression task, we have visually and statistically analysed scatter plots for the feature-grade relations. From the scatter plots it seems that there is a negative correlation between the grade and `reviewers`, `comment ratio` and `churn` features. The values of these features are less often high for the better graded examples and are less often low for the lower graded examples. Performed statistical tests showed that there is statistically significant difference in means of these three features when split into two groups according to the grade.

The better graded changes have an average number of reviewers 3.6 as compared to 5.8 for the lower graded changes. This difference could be caused by the poor development performance requiring more attention from the reviewers to keep the code quality good. Similar correlation can be observed on the `comment ratio` and `churn` features. The difference between average churn of better graded and lower graded changes is almost 800 lines. As this metric sums all the changed lines after the first patchsets, the changes with a higher number of patchsets with extensive changes have also a higher churn value. Therefore the expectation that more patchsets or great amounts of code modifications in later patchsets indicate a worse development performance seems to be supported by the great difference in means of the `churn` feature. Furthermore, the average `comment ratio` value for the lower graded changes is 0.5587, which means that on average the lower graded changes have a greater number of comments than the half of overall code lines changed. Even though this metric is also dependent on the `size` feature, and therefore might have some limitations in explanatory power, it seems that it supports the expectation that the code produced during a worse development needs more comments per each produced line.

To further investigate the significance of each feature on the performance evaluation prediction we have generated a classification tree. After learning and pruning procedures it remained with the following five features in the decision nodes: `contributors, reviewers, size, comment ratio` and `churn`. Although the accuracy of the decision tree slightly above 75% limits the reliability of the tree analysis, it provides some additional information for the feature analysis.

All three features identified by the statistical analysis have been used by the decision tree as well, which supports their relevance towards the performance evaluation prediction. The tree algorithm has also used `contributors` and `size` features. However, both of these features lie in the decision tree structure on lower decision nodes and also the statistical analysis has not found a direct relation to the grade values.

5.3 Model accuracy (RQ3)

To assess the goodness of fit of learned models we have calculated the R^2 and adjusted-R^2 values. The R^2 values represents how well the model explains the variability of the data around its mean. A R^2 value close to 1 symbolizes almost perfect fit, while R^2 value close to 0 describes worse fitting models. Although a very high R^2 value can be in some cases undesirable, in the value prediction case a higher R^2 value is a good sign. The achieved value 0.5393 explains more than 50% of the data variance, but for label prediction is below a desirable value.

The adjusted-R^2 value of 0.4385 is lower than the R^2 value by approximately 0.1. The adjusted-R^2 value is modified by the number of explanatory values used in the model. Therefore the observed difference between the R^2 values can be caused by using features that are not useful for the grade values prediction. Removing features that have a small contribution to the performance evaluation could lead to a better adjusted-R^2 value. Furthermore, introducing new features that have more explanatory power for the performance evaluation approximation, could lead to overall improvement of the performance.

Finally, to cross-validate the achieved regression accuracy we have selected and labelled a sample of unlabelled data and sent it for label validation to the review experts. Results show that 73.68% of the predicted labels were considered correct. Although there is still a lack of reliability in the prediction, the accuracy is close to 75%, which gives a solid base for further improvements. Therefore we believe that achieved results show that it is possible to approximate the review experts' performance evaluation function by machine learning methods using the data from code reviews.

5.4 Threats to validity

5.4.1 Threats to internal validity

The internal validity of our research is limited by the size of labelled sample. The labelled sample lacks size to have a good explanatory power for the performed statistical analysis as well as to provide a solid and varying base for the machine learning algorithms. Also, the obtained labels come from two different review experts, which can introduce some inconsistency in the evaluation metric interpretation by the individual review experts. To mitigate the impact of insufficient size of the labelled sample we have tried to select as representative sample as possible with regard to mentioned constraints.

Another limitation is that the labelled data is related to two particular review experts. Although these are responsible for reviewing large parts of the project, this limitation might exclude some changes relevant to the performance evaluation task and lowers the possible generalisation of results within the project. Then, the achieved accuracy from the manual validation is based on a random sample of limited size and is also a subject to the validation bias of review experts.

Furthermore, we have selected the features and data preprocessing methods based on our knowledge, knowledge of the review experts and the knowledge acquired from related research. However, results of the machine learning algorithms are strongly dependent on the quality of data representation and therefore the resulting performance of models might be limited by the particular choices made.

Finally, the interpretation of results is based on the knowledge of the researcher. There might exist other interpretations of achieved R^2 or adjusted-R^2 values as well as the results of statistical analysis.

5.4.2 Threats to external validity

A major threat to external validity is the fact that the available data comes from only one particular Ericsson's project. A good generalisation of results to other projects is not guaranteed. However, as the data comes from a standardised review process within the Gerrit environment, generalising the outcomes to other projects using one of the modern code review tools can be possible to some extent.

Another threat to external validity is the fact that we have used one particular semi-supervised method in combination with one particular regression algorithm. To provide a more robust answers to our research questions different combinations of algorithms should be tested and they should be tested on more datasets.

Chapter 6

Conclusions and Future Work

The final chapter draws conclusions, discusses the influence of this work on current body of knowledge and proposes directions for the future work.

6.1 Conclusions

This thesis aimed to investigate possibilities of automatically evaluating the quality of a development process based on data from modern code review tools using machine learning tools. At the same time it aimed to compare the performance of semi-supervised learning and supervised learning on this task. To achieve the declared aims we have designed a set of features for the performance evaluation task, gathered data, selected a semi-supervised learning method, performed an experiment to compare its performance, statistically analysed the feature set and finally additionally validated the accuracy of learned models with the help of Ericsson's review experts.

The obtained results did not support the expectation that the semi-supervised learning method would improve the performance of the regression algorithm. Nevertheless, as the results were similar for both methods we can conclude that in our particular case the semi-supervised method did not improve nor degrade the performance. Advantages of semi-supervised learning might be limited due to the nature of data or insufficient size of the labelled data sample.

The analysis of feature set has shown that at least three of the designed features have some direct relation to the performance evaluation. It seems that a worse development process attracts more attention from the reviewers, leads to a greater ratio of comments number to the change size and also is linked to a greater amount of changes required after the first patchset. We believe that especially the amount of changes after the first patchset is a promising metric that should be investigated more deeply, as it seems to have a visible negative correlation with the performance evaluation grade. The analysis also suggests that some of the features from our feature set might not have a strong explanatory power for the problem and their refinement could bring improvements in the final accuracy of models.

Although the achieved accuracy of models reaching 73.68% is not completely sufficient for a reliable and robust prediction, we believe it proves the feasibility of using the machine learning for the performance evaluation approximation considering the limitations of data size.

Finally, to wrap our findings we answer our research question in the following way.

RQ1 We have not managed to prove that the metric based regularisation semi-supervised learning approach would be beneficial for the approximation of the performance evaluation.

RQ2 We have found that the number of reviewers, comment ratio and the amount of changes code lines after the first patchset are with high probability relevant for the performance evaluation task.

RQ3 We have managed to achieve 73.68% accuracy of performance evaluation on previously unseen data and therefore we believe that we have proved that it is possible to approximate an expert's performance evaluation function using machine learning methods on the data from code reviews.

6.2 Future work

We believe that our work has brought findings worth a deeper investigation. For the future work we propose a few directions of research focus.

It seems that some of the used features lack an useful explanatory power. Therefore we think that deeper analysis of features and their explanatory power with the use of additional tools, e.g. $AkaikeInformationCriterion$ (AIC) [10], could bring significant improvements in the quality of regression analysis.

Our thesis has used only one particular combination of semi-supervised learning method and regression algorithm. Future work could investigate the performance of different regression algorithms using the same metric based regularisation method and also could compare the influence on performance of other semi-supervised methods e.g. Co-Training with k-nearest neighbours learning method as proposed by Zhou et al. [37].

Additionally, future research could focus on obtaining similar performance evaluation labels for code review data of other projects and replicating the experiment and feature analysis on a different project to further verify findings presented in this work.

References

[1] Chromium. https://www.chromium.org/, 2016. [Online; accessed 5-April-2016].

[2] Gerrit. Web based code review and project management for Git based projects. https://code.google.com/p/gerrit/, 2016. [Online; accessed 5-April-2016].

[3] Git. https://git-scm.com/, 2016. [Online; accessed 5-April-2016].

[4] Mysql. https://www.mysql.com/, 2016. [Online; accessed 5-April-2016].

[5] Qt. https://www.qt.io/, 2016. [Online; accessed 5-April-2016].

[6] The r project. https://www.r-project.org/, 2016. [Online; accessed 5-April-2016].

[7] Rietveld. https://github.com/rietveld-codereview/rietveld, 2016. [Online; accessed 5-April-2016].

[8] Weka. http://www.cs.waikato.ac.nz/ml/weka/, 2016. [Online; accessed 5-April-2016].

[9] Yaser S Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272:64–69, 1995.

[10] Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[11] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

[12] Martin D Buhmann. Radial basis functions: theory and implementations. *Cambridge monographs on applied and computational mathematics*, 12:147–165, 2004.

[13] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[14] Vladimir Cherkassky and Filip M Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[15] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. Wiley, 2001.

[16] M.E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 1976.

[17] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge univ. press, 2012.

[18] Stanton A Glantz and Bryan K Slinker. *Primer of applied regression and analysis of variance*. McGraw-Hill, Health Professions Division, 1990.

[19] Romans Grisins and Andrej Sekáč. Influence of the reviewers' role and workload on defect types found during code reviews. unpublished, 2015.

[20] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[21] Pilsung Kang, Dongil Kim, and Sungzoon Cho. Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing. *Expert Systems with Applications*, 51:85–106, 2016.

[22] Andrew Koenig. Patterns and antipatterns. *The patterns handbook: techniques, strategies, and applications*, 13:383, 1998.

[23] A. G. Koru, D. Zhang, K. El Emam, and H. Liu. An investigation into the functional form of the size-defect relationship for software modules. *IEEE Transactions on Software Engineering*, 35(2):293–304, March 2009.

[24] Raula Gaikovina Kula, Ana Erika Camargo Cruz, Norihiro Yoshida, Kazuki Hamasaki, Koji Fujiwara, Xu Yang, and Hiroyuki Iida. Using profiling metrics to categorise peer review types in the android project. In *Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on*, pages 146–151. IEEE, 2012.

[25] JunWei Liang and Osamu Mizuno. Analyzing involvements of reviewers through mining a code review repository. In *Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pages 126–132. IEEE, 2011.

[26] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E. Hassan. The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 192–201, New York, NY, USA, 2014. ACM.

[27] Andrew Meneely, Alberto C. Rodriguez Tejeda, Brian Spates, Shannon Trudeau, Danielle Neuberger, Katherine Whitlock, Christopher Ketant, and Kayla Davis. An empirical investigation of socio-technical code review metrics and security vulnerabilities. In *Proceedings of the 6th International Workshop on Social Software Engineering*, SSE 2014, pages 37–44, New York, NY, USA, 2014. ACM.

[28] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[29] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2015.

[30] Rodrigo Morales, Shane McIntosh, and Foutse Khomh. Do code review practices impact design quality? a case study of the qt, vtk, and itk projects. pages 171–180. IEEE, 2015.

[31] Mark JL Orr et al. Introduction to radial basis function networks, 1996.

[32] Dale Schuurmans and Finnegan Southey. Metric-based methods for adaptive model selection and regularization. *Machine Learning*, 48(1):51–84, 2002.

[33] Patrick Stalph. *Analysis and design of machine learning techniques: evolutionary solutions for regression, prediction, and control problems*. Springer Science & Business Media, 2014.

[34] Toshifumi Tanaka, Keishi Sakamoto, Shinji Kusumoto, Ken-ichi Matsumoto, and Tohru Kikuno. Improvement of software process by process description and benefit estimation. In *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pages 123–123. IEEE, 1995.

[35] Ian H. Witten and Eibe Frank. *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[36] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of Biostatistics*, 1998.

[37] Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *IJCAI*, volume 5, pages 908–913, 2005.

[38] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.