Using Source Code Metrics to Predict Change-Prone Web Services: A Case-Study on eBay Services

Lov Kumar\textsuperscript{1} Santanu Kumar Rath\textsuperscript{1} Ashish Sureka\textsuperscript{2}

\textsuperscript{1}NIT Rourkela, India
Email: lovkumar505@gmail.com, skrath@nitrkl.ac.in

\textsuperscript{2}ABB India, India
Email: ashish.sureka@in.abb.com

MaLTeSQuE 2017
Table of Contents

1 Research Motivation, Aim, Contributions
   - Objectives and Context Setting
   - Related Work
   - Research Contributions

2 Experimental Dataset and Setup
   - eBay Trading API
   - Data Collection and Preparation

3 Research Questions

4 Empirical Validation of Software Metrics
   - Source Code Metrics Validation
   - Least Square Support Vector Machine (LSSVM) Classifier
   - Comparison of Results

5 Answers to Research Questions

6 Conclusion

7 References
Maintainability Prediction - Service Oriented Software

Change-Prone Class Prediction

Focus *preventive maintenance efforts* such as testing, peer-reviews and source code refactoring on change-prone regions [7][10]

Web Services

Distributed web application components, implemented in different languages, deployed on different client/server platforms, represented by interfaces, communicate using open protocols [5][12]

WSDL (Web Service Description Language)

An XML format and provides users of the web service with a point of contact [5]
Change proneness prediction for web service interfaces

Service oriented systems is a different programming paradigm in comparison to object-oriented applications

Romano et al. mention that there are lack of studies that examine indicators of changes for service-oriented system [14]

Specific Research Aim

Can source code metrics be used to predict change-proneness of web services?
Table of Contents

1 Research Motivation, Aim, Contributions
   • Objectives and Context Setting
   • Related Work
   • Research Contributions

2 Experimental Dataset and Setup
   • eBay Trading API
   • Data Collection and Preparation

3 Research Questions

4 Empirical Validation of Software Metrics
   • Source Code Metrics Validation
   • Least Square Support Vector Machine (LSSVM) Classifier
   • Comparison of Results

5 Answers to Research Questions

6 Conclusion

7 References
Coscia et al. [4]
High correlation between several traditional (source code-level) OO metrics and the catalog of (WSDL-level) service metrics [4].

Lov et al. [8]
Quality of Service (QoS) characteristics of web services have a correlation with several source code metrics and hence can be estimated by analyzing the source code [8].

Several interface complexity attributes such as the data, relation, format, structure, data flow and language are correlated with source code implementing the web-services.
Research Motivation, Aim, Contributions
Objectives and Context Setting
Related Work
Research Contributions

Experimental Dataset and Setup
eBay Trading API
Data Collection and Preparation

Research Questions

Empirical Validation of Software Metrics
Source Code Metrics Validation
Least Square Support Vector Machine (LSSVM) Classifier
Comparison of Results

Answers to Research Questions

Conclusion

References

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
Using Source Code Metrics to Predict Change-Prone Web Services
Novel and Unique Contributions

Contribution on Solution Approach and Evaluation

First study on using source code metrics (implementing a web service) to predict change-prone web service interfaces (defined using WSDL) using kernel based learning techniques - Least Squares Support Vector Machines (LS-SVM).

Empirical evaluation of the proposed approach on real world and publicly available dataset consisting of several version of eBay services.
# Table of Contents

1. Research Motivation, Aim, Contributions
   - Objectives and Context Setting
   - Related Work
   - Research Contributions

2. Experimental Dataset and Setup
   - eBay Trading API
   - Data Collection and Preparation

3. Research Questions

4. Empirical Validation of Software Metrics
   - Source Code Metrics Validation
   - Least Square Support Vector Machine (LSSVM) Classifier
   - Comparison of Results

5. Answers to Research Questions

6. Conclusion

7. References
We download eBay web services from publicly available source\(^a\) - our experiments can be replicated, used for benchmarking & comparison.

The eBay Trading API is an active project and consists of several versions. The latest version as of 17 June 2016 is 973.

# Table of Contents

1. Research Motivation, Aim, Contributions  
   - Objectives and Context Setting  
   - Related Work  
   - Research Contributions

2. Experimental Dataset and Setup  
   - eBay Trading API  
   - Data Collection and Preparation

3. Research Questions

4. Empirical Validation of Software Metrics  
   - Source Code Metrics Validation  
   - Least Square Support Vector Machine (LSSVM) Classifier  
   - Comparison of Results

5. Answers to Research Questions

6. Conclusion

7. References
Multi-step Process of data Collection and Analysis

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Using Source Code Metrics to Predict Change-Prone Web Services
We compute 21 source code metrics CKJM extended\(^a\).

We use WSDL Diff tool for comparing subsequent versions of WSDL interfaces [3]. The WSDL Diff tool automatically extracts the changes in subsequent versions of WSDL interfaces.

We use the wsimport\(^b\) tool to parse WSDL document file of a Web Service and generate its corresponding Java class (extracting the Java source code implementing the service).

\(^a\)http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/
\(^b\)http://docs.oracle.com/javase/6/docs/technotes/tools/share/wsimport.html
The Java classes which contain changed (addition, deletion and modification) elements of the Web Service such as (Operation, Message, XSDType) are termed as changed classes.

The percentage of change classes from version 865 to 867 is 15.75 and the percentage of change classes from 869 to 871 is 7.89.

We consider all the changes as equivalent and do not differentiate between type of changes (a bug fix for a feature enhancement or refactoring) or the size of the change.
Two Research Questions

**RQ1:** Is it possible to predict change-proneness of web-services (defined using a WSDL document) using source code metrics implementing the web services? What is the performance of 21 CKJM metrics in-terms of their predictive power?

**RQ2:** What is the variation in performance (measured in-terms of accuracy and F-measure) of several LS-SVM classification models over different set of source code metrics? Do we see different performance for different (linear, polynomial and RBF) LS-SVM kernel functions?
# Table of Contents

1. Research Motivation, Aim, Contributions
   - Objectives and Context Setting
   - Related Work
   - Research Contributions

2. Experimental Dataset and Setup
   - eBay Trading API
   - Data Collection and Preparation

3. Research Questions

4. **Empirical Validation of Software Metrics**
   - Source Code Metrics Validation
   - Least Square Support Vector Machine (LSSVM) Classifier
   - Comparison of Results

5. Answers to Research Questions

6. Conclusion

7. References
Source code metrics - SM or Selected Metrics

1. **Star (∗):** source code metrics selected after t-test analysis.
2. **Circle with star (⊙):** source code metrics selected after t-test and ULR analysis.
3. **Square with circle and star (□):** source code metrics selected after t-test, ULR analysis and cross correlation analysis.
4. **Hexagonal with square, circle and star (■):** source code metrics selected after t-test, ULR analysis, cross correlation analysis and MLR stepwise forward selection method.
Source Code Metrics Validation

(a) Selected Set of Metrics

(b) Hypothesis

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Using Source Code Metrics to Predict Change-Prone Web Services
t-test Analysis

First step of source code metrics validation is to test the statistical significance between change and non-change-proneness group source code metric.

We apply $t$–test on each source code metric and consider $p$–value of each source code metrics.

The metrics having $p$–value lesser than 0.05 have strong discriminatory power.

WMC, DIT, CBO, RFC, Ca, NPM, MOA, CAM, IC, CBM, and AMC source code metrics significantly differentiate the change or non-change-proneness classes for eBay version 863.
Source Code Metrics Validation

Univariate Logistic Regression (ULR) Analysis

Investigate whether the selected set of source code metrics using \( t - \)test analysis are significant predictors of change-proneness classes or not.

A source code metrics is significant predictor of class change-proneness if its p-value of coefficient is less than 0.05.

WMC, DIT, CBO, RFC, Ca, NPM, MOA, CAM, IC, CBM, AMC only WMC, CBO, Ca, NPM, CAM, IC, CBM, AMC source code metrics are significant predictors of change-proneness for eBay web service version 863.
Correlation Analysis between Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>WMC</th>
<th>DIT</th>
<th>NOC</th>
<th>CBO</th>
<th>RFC</th>
<th>LCOM</th>
<th>Ca</th>
<th>Ce</th>
<th>NPM</th>
<th>LCOM3</th>
<th>DAM</th>
<th>MOA</th>
<th>MFA</th>
<th>CAM</th>
<th>IC</th>
<th>CBM</th>
<th>AMC</th>
<th>LOC</th>
<th>SLOC-P</th>
<th>SLOC-L</th>
<th>MVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DIT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CBO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RFC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LCOM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ca</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ce</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NPM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LCOM3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DAM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MFA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CAM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>IC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CBM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AMC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LOC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SLOC-P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SLOC-L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MVG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Correlation Analysis between Metrics - Symbols

1. **Black Circle (●):** r value between 0.7 and 1.0 indicate a strong positive linear relationship.

2. **White circle (○):** r value between 0.3 and 0.7 indicate a weak positive linear relationship.

3. **Black Diamond ( ):** r value between -1 and -0.7 indicate a strong negative linear relationship.

4. **White Diamond (◇):** r value between -0.7 and -0.3 indicate a weak negative linear relationship.

5. **Blank Circle:** no linear relationship.
Multivariate Linear Regression Stepwise Forward Selection:

The selected set of source code metrics obtained after cross correlation analysis does necessarily mean that we have a suitable set of source code metrics for change-proneness model development.

Multivariate linear regression stepwise forward selection method is considered to select right set of source code metrics for development of change-proneness models.
## Table of Contents

1. Research Motivation, Aim, Contributions
   - Objectives and Context Setting
   - Related Work
   - Research Contributions

2. Experimental Dataset and Setup
   - eBay Trading API
   - Data Collection and Preparation

3. Research Questions

4. **Empirical Validation of Software Metrics**
   - Source Code Metrics Validation
   - Least Square Support Vector Machine (LSSVM) Classifier
   - Comparison of Results

5. Answers to Research Questions

6. Conclusion

7. References

---

**Lov Kumar, Santanu Kumar Rath, Ashish Sureka**

Using Source Code Metrics to Predict Change-Prone Web Services
Least Square Support Vector Machine (LSSVM) Classifier

**LSSVM**

LSSVM classifier with three different kernel methods for validating the selected set of source code metrics.

20 fold cross-validation for comparing the predictive models.

The performance of each prediction model is evaluated in terms of two performance parameters i.e., **accuracy** and **F-Measure**.
Performance Parameters

**Figure:** Validation of prediction models constructed using selected subset of metrics and all metrics (Linear Kernel)
Figure: Results of the validation of prediction models constructed using selected subset of metrics and all metrics (Polynomial Kernel)
**Figure**: Results of the validation of prediction models constructed using selected subset of metrics and all metrics (RBF Kernel)
Performance Parameters

1. The y-axis represents the accuracy and f-measure performance evaluation metrics.
2. The two bars represent the performance for the AM and SM metrics respectively.
3. Performance values for different versions of eBay web service using LSSVM with linear, polynomial and radial basic function is shown in previous figures.
Observations

Takeaways from Experiments

We observe that change-proneness model developed using selected set of source code metrics passes the desired prediction accuracy and comparable with the models that are build by considering all twenty one metrics.

This results confirm that the ability of these selected source code metrics to predict change-proneness in the eBay web service.
Table of Contents

1. Research Motivation, Aim, Contributions
   • Objectives and Context Setting
   • Related Work
   • Research Contributions

2. Experimental Dataset and Setup
   • eBay Trading API
   • Data Collection and Preparation

3. Research Questions

4. Empirical Validation of Software Metrics
   • Source Code Metrics Validation
   • Least Square Support Vector Machine (LSSVM) Classifier
   • Comparison of Results

5. Answers to Research Questions

6. Conclusion

7. References
### Table: Feature Selection Techniques

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P-value</td>
<td>t-value</td>
</tr>
<tr>
<td></td>
<td>AM</td>
<td>SM</td>
</tr>
<tr>
<td>AM</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td>SM</td>
<td>0.80</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>P-value</td>
<td>t-value</td>
</tr>
<tr>
<td></td>
<td>AM</td>
<td>SM</td>
</tr>
<tr>
<td>AM</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>SM</td>
<td>0.99</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Two sets (one for each performance measure) are used, each with 15 data points (3 classifier * 5 dataset).

We observe that there is no significant difference between these approaches due to the fact that p-value greater than 0.05.

Based on value of mean difference, we observe that the model developed using all metrics resulted in sightly better (only 0.57 % higher accuracy) results compared to models using selected set of metrics.
### Table: Classification Methods

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P-value</td>
<td>t-value</td>
</tr>
<tr>
<td></td>
<td>Lin</td>
<td>Poly</td>
</tr>
<tr>
<td>Lin</td>
<td>1.00</td>
<td>0.56</td>
</tr>
<tr>
<td>Poly</td>
<td>0.56</td>
<td>1.00</td>
</tr>
<tr>
<td>RBF</td>
<td>0.97</td>
<td>0.60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P-value</th>
<th>t-value</th>
<th>Mean Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lin</td>
<td>1.00</td>
<td>0.41</td>
<td>0.72</td>
</tr>
<tr>
<td>Poly</td>
<td>0.41</td>
<td>1.00</td>
<td>0.59</td>
</tr>
<tr>
<td>RBF</td>
<td>0.72</td>
<td>0.59</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Kernel Methods

LSSVM method with three different kernel methods are considered to develop a model to predict class change-proneness.

We observe that there is no significant difference between these approaches, due to the fact that the p-value is greater than 0.05.

However, upon closely inspecting the value of mean difference, LSSVM with linear kernel function yields better results compared to other kernels.
We infer that it is possible to predict change-proneness of web services (defined using a WSDL document) using source code metrics implementing the web services.

The accuracy of the predictive models are above 80% for all the three types of Kernel and for all the five version of the metrics.

A consistant accuracy of above 80% for both the metrics set (AM and SM) is an evidence of the effectiveness of the proposed approach.
We conclude that the performance of the LSSVM method varies with the different set of source code metrics.

Selection of classification metrics to develop a model for predicting change-proneness classes is affected by the selection of source code metrics.

AM metrics outperform the SM metric for the linear kernel and RBF kernel.

The AM metric does not dominate SM metric for the polynomial kernel.
Conclusion

It is possible to predict change proneness of WSDL documents and services using source code metrics and kernel based learning techniques.

The model developed using all metrics results in slightly better (only 0.57% higher accuracy) results compared to models on selected set of metrics.

The predictive model developed using **LS-SVM linear kernel** yields better results compared to other kernels.
The performance of the selected set of source code metrics varies with the different classification methods (such as linear, polynomial and RBF kernel).

Even after removing 85.71% (Average) of the available number of source code metrics, the developed change-proneness prediction models were not adversely affected; in fact, in some of the cases the results were better.
References I


References II


References III


