Empirical Analysis on Effectiveness of Source Code Metrics for Predicting Change-Proneness

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

ISEC 2017
Table of Contents

1. Research Motivation and Aim
   - Research Motivation
   - Research Aim

2. Related work

3. Experimental Dataset
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4. Feature Selection Methods
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5. Research Method

6. Model Building Results

7. Conclusion

8. References
Change-prone Class Prediction

Change-prone classes or modules are defined as software components in the source code which are likely to change in the future.

Prediction and early identification of change-prone components are useful for test-resource optimization.

Source code metrics are used to measure the internal structure of software system such as complexity, coupling, cohesion, inheritance, and size.

Motivation: Source code metrics and machine learning based technique for change-proneness prediction.
Four Source Code Metrics Dimensions

7 size metrics, 18 cohesion metrics, 20 coupling metrics, and 17 inheritance metrics

Selection of the suitable set of source code metrics is an integral component of the predictive model development process

Motivation: Investigate five different types of features selection techniques such as univariate logistic regression analysis, gain ratio feature evaluation, information gain feature evaluation, principal component analysis (PCA), and rough set analysis (RSA)
# Table of Contents

1. **Research Motivation and Aim**
   - Research Motivation
   - Research Aim

2. Related work

3. **Experimental Dataset**
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4. **Feature Selection Methods**
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5. Research Method

6. Model Building Results

7. Conclusion

8. References
Objective [Metrics and Feature Selection]

[1] Investigate the performance of four different source code metrics dimensions such as size metrics (7), cohesion metrics (18), coupling metrics (20), and inheritance metrics (17) for change-proneness prediction.

[2] Investigate the performance of five different types of features selection techniques such as univariate logistic regression analysis, gain ratio feature evaluation, information gain feature evaluation, principal component analysis (PCA), and rough set analysis (RSA) to select right set of source code metrics.
Objective [Learning Algorithms and Ensemble Methods]

[3] Investigate the performance of eight different learning algorithms: LOGR, NBC, ELM-LIN, ELM-PLY, ELM-RBF, SVM-LIN, SVM-RBF, SVM-SIG to develop a model to predict change-proneness of OO software.

[4] Investigate the performance of two different types of ensemble methods to come up with better performance as compared to the individual models.
Literature Survey

**Henry and Kafura [102]**

Henry and Kafura considered correlation analysis for measuring the correlation between changeability i.e., number of changed source lines in the Unix operating system and source code metrics [102]. They found that the source code metrics are strongly correlated with changeability. They defined these source code metrics using information flow among the system components.

**Ruchika Malhotra and Anuradha Chug [107]**

Ruchika Malhotra and Anuradha Chug studied the relationship between object oriented metrics and maintainability i.e., changeability of software [107].
Hongmin Lu et al. [105]
Hongmin Lu et al. considered statistical meta-analysis techniques to investigate the ability of sixty Object-Oriented source code metrics to predict change-proneness [105]

Yuming Zhou and Hareton Leung [164]
Yuming Zhou and Hareton Leung considered multiple adaptive regression splines (MARS) modeling technique to build software maintainability prediction models using the software metrics [164]
We use two different versions of **Eclipse software application** (Eclipse Version 2.0 \(^a\) and Eclipse Version 2.1 \(^b\))

Eclipse is a long-running, widely-used, publicly available, large and complex open-source project


Eclipse Software Application

We compute the source code metrics value for only those Java file which appear in both the versions i.e., Eclipse 2.0 and Eclipse 2.1. We use Perl tool to compute the change-proneness module between two version of Eclipse software \(^a\) [105]

We eliminate several Java files (Refer to paper for the procedure followed to eliminate files)

\(^a\)https://www.perl.org/get.html
## Table of Contents

1. Research Motivation and Aim
   - Research Motivation
   - Research Aim

2. Related work

3. **Experimental Dataset**
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4. Feature Selection Methods
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5. Research Method

6. Model Building Results

7. Conclusion

8. References
### Cohesion: From COA to TCC

<table>
<thead>
<tr>
<th>Metric</th>
<th>Max</th>
<th>Min</th>
<th>Mean</th>
<th>Median</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA</td>
<td>110</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
<td>3.636</td>
</tr>
<tr>
<td>CAMC</td>
<td>1</td>
<td>0.014</td>
<td>0.221</td>
<td>0.179</td>
<td>0.15</td>
</tr>
<tr>
<td>CO</td>
<td>1</td>
<td>-2</td>
<td>0.122</td>
<td>0.089</td>
<td>0.461</td>
</tr>
<tr>
<td>DCD</td>
<td>1</td>
<td>0</td>
<td>0.54</td>
<td>0.5</td>
<td>0.379</td>
</tr>
<tr>
<td>DCI</td>
<td>1</td>
<td>0</td>
<td>0.617</td>
<td>0.714</td>
<td>0.391</td>
</tr>
<tr>
<td>ICH</td>
<td>2976</td>
<td>0</td>
<td>21.051</td>
<td>4</td>
<td>85.856</td>
</tr>
<tr>
<td>LCC</td>
<td>1</td>
<td>0</td>
<td>0.604</td>
<td>0.672</td>
<td>0.397</td>
</tr>
<tr>
<td>LCOM1</td>
<td>171850</td>
<td>0</td>
<td>217.239</td>
<td>23</td>
<td>3018.651</td>
</tr>
<tr>
<td>LCOM2</td>
<td>166390</td>
<td>0</td>
<td>185.78</td>
<td>13</td>
<td>2893.3</td>
</tr>
<tr>
<td>LCOM3</td>
<td>492</td>
<td>1</td>
<td>6.78</td>
<td>4</td>
<td>12.283</td>
</tr>
<tr>
<td>LCOM4</td>
<td>282</td>
<td>1</td>
<td>3.236</td>
<td>2</td>
<td>6.176</td>
</tr>
<tr>
<td>LCOM5</td>
<td>2</td>
<td>0</td>
<td>0.771</td>
<td>0.833</td>
<td>0.28</td>
</tr>
<tr>
<td>NHD</td>
<td>1</td>
<td>0</td>
<td>0.651</td>
<td>0.69</td>
<td>0.205</td>
</tr>
<tr>
<td>NEWCO</td>
<td>1</td>
<td>0</td>
<td>0.369</td>
<td>0.306</td>
<td>0.274</td>
</tr>
<tr>
<td>NEWLCOM5</td>
<td>1</td>
<td>0</td>
<td>0.333</td>
<td>0.267</td>
<td>0.243</td>
</tr>
<tr>
<td>OCC</td>
<td>1</td>
<td>0</td>
<td>0.655</td>
<td>0.714</td>
<td>0.31</td>
</tr>
<tr>
<td>PCC</td>
<td>1</td>
<td>0</td>
<td>0.534</td>
<td>0.563</td>
<td>0.332</td>
</tr>
<tr>
<td>SNHD</td>
<td>1</td>
<td>-1</td>
<td>-0.187</td>
<td>0</td>
<td>0.527</td>
</tr>
<tr>
<td>TCC</td>
<td>1</td>
<td>0</td>
<td>0.53</td>
<td>0.5</td>
<td>0.383</td>
</tr>
</tbody>
</table>
### Inheritance: From AID to SPD

<table>
<thead>
<tr>
<th>Metric</th>
<th>Max</th>
<th>Min</th>
<th>Mean</th>
<th>Median</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>AID</td>
<td>7</td>
<td>0</td>
<td>1.265</td>
<td>1</td>
<td>1.343</td>
</tr>
<tr>
<td>CLD</td>
<td>7</td>
<td>0</td>
<td>0.309</td>
<td>0</td>
<td>0.776</td>
</tr>
<tr>
<td>DIT</td>
<td>7</td>
<td>0</td>
<td>1.265</td>
<td>1</td>
<td>1.343</td>
</tr>
<tr>
<td>DP</td>
<td>420</td>
<td>0</td>
<td>4.82</td>
<td>1</td>
<td>16.778</td>
</tr>
<tr>
<td>DPA</td>
<td>129</td>
<td>0</td>
<td>2.286</td>
<td>0</td>
<td>5.233</td>
</tr>
<tr>
<td>DPD</td>
<td>415</td>
<td>0</td>
<td>2.534</td>
<td>0</td>
<td>15.874</td>
</tr>
<tr>
<td>NMA</td>
<td>596</td>
<td>0</td>
<td>11.429</td>
<td>7</td>
<td>18.455</td>
</tr>
<tr>
<td>NMI</td>
<td>596</td>
<td>0</td>
<td>23.809</td>
<td>6</td>
<td>42.686</td>
</tr>
<tr>
<td>NMO</td>
<td>124</td>
<td>0</td>
<td>1.988</td>
<td>0</td>
<td>4.316</td>
</tr>
<tr>
<td>NOA</td>
<td>7</td>
<td>0</td>
<td>1.265</td>
<td>1</td>
<td>1.343</td>
</tr>
<tr>
<td>NOC</td>
<td>155</td>
<td>0</td>
<td>0.739</td>
<td>0</td>
<td>3.967</td>
</tr>
<tr>
<td>NOPD</td>
<td>381</td>
<td>0</td>
<td>1.606</td>
<td>0</td>
<td>10.654</td>
</tr>
<tr>
<td>NOP</td>
<td>1</td>
<td>0</td>
<td>0.626</td>
<td>1</td>
<td>0.484</td>
</tr>
<tr>
<td>SIX</td>
<td>1.596</td>
<td>0</td>
<td>0.094</td>
<td>0</td>
<td>0.156</td>
</tr>
<tr>
<td>SP</td>
<td>110</td>
<td>0</td>
<td>0.665</td>
<td>0</td>
<td>3.74</td>
</tr>
<tr>
<td>SPA</td>
<td>10</td>
<td>0</td>
<td>0.264</td>
<td>0</td>
<td>0.742</td>
</tr>
<tr>
<td>SPD</td>
<td>110</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
<td>3.636</td>
</tr>
</tbody>
</table>
Source Code Metrics

**Coupling**

The information flow among various program components in the object oriented software implementation is measured using coupling and there are several metrics such as Data Abstraction Coupling (DAC) and Coupling between Objects (CBO) to measure coupling [127]

**Cohesion**

Cohesion is defined as the degree or extent to which various elements in a design unit such as packages and classes are related to each other and is measured using several metrics such as Lack of Cohesion in Methods (LCOM) and Information-Flow based Cohesion (ICH) [110]
Source Code Metrics

Inheritance

Inheritance metrics such as Average Inheritance Depth (AID) and Class-to-Leaf Depth (CLD) metric measure quality and complexity of class inheritance hierarchies [144]

Size

Size metrics such as (Lines of Code) LOC determine the size of the program code [103]
Observations

Results reveals substantial variation or dispersion in the values of 62 source code metrics which shows wide variability in the structure and size of program code and elements

Source Lines of Code (SLOC) value varies from a minimum of 4 to a maximum of 3669

The mean value of 115.431 for SLOC means that a large number of Classes have SLOC more than 100.
Correlation Matrix between Source Code Metrics

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Empirical Analysis on Effectiveness of Source Code Metrics for Predicting Change-Proneness
Zooming a Section of Correlation Matrix
Table of Contents

1 Research Motivation and Aim
   • Research Motivation
   • Research Aim

2 Related work

3 Experimental Dataset
   • Source Code Metrics
   • Correlation Analysis
   • Effectiveness of Metrics

4 Feature Selection Methods
   • Univariate Logistic Regression (ULR)
   • Principal Component Analysis (PCA)
   • Gain Ratio (GR) and Information Gain (IG)

5 Research Method

6 Model Building Results

7 Conclusion

8 References
Method

We compute the association between 62 metrics consisting of dependent and independent variables using the Pearson’s correlations coefficient ($r$)

A Black circle represents an $r$ value between 0.7 and 1.0 or between $-0.7$ and $-1.0$ indicating a strong positive or negative linear relationship respectively.

A White circle $r$ value between 0.3 and 0.7 or $-0.3$ and $-0.7$ indicating a weak positive or negative linear relationship respectively.
Metrics Correlation Analysis

Observations

A blank cell represents no linear relationships between the two variables.

There is a strong positive linear relationship between LCOM1 and seven other variables LCOM2, LCOM3, LCOM4, NMA, NAIMP, NUMPA and SLOC.

A weak linear relationship between CAMC and DCD as well as CBO and ICH.

Metrics such as NA, NAIMP, NM, NMIMP and NUMPA are part of the size metrics and they have strong as well as weak correlations with several coupling metrics OCAEC, NIHICP, MPC and IHICP.
Table of Contents

1 Research Motivation and Aim
   - Research Motivation
   - Research Aim

2 Related work

3 Experimental Dataset
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4 Feature Selection Methods
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5 Research Method

6 Model Building Results

7 Conclusion

8 References
## Metrics Set

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Dependent Variable</th>
<th>Independent Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Change-proneness</td>
<td>All metrics</td>
</tr>
<tr>
<td>A2</td>
<td>Change-proneness</td>
<td>Cohesion metrics</td>
</tr>
<tr>
<td>A3</td>
<td>Change-proneness</td>
<td>Coupling metrics</td>
</tr>
<tr>
<td>A4</td>
<td>Change-proneness</td>
<td>Size metrics</td>
</tr>
<tr>
<td>A5</td>
<td>Change-proneness</td>
<td>Inheritance metrics</td>
</tr>
<tr>
<td>A6</td>
<td>Change-proneness</td>
<td>Selected set of metrics using gain ratio feature (GRS)</td>
</tr>
<tr>
<td>A7</td>
<td>Change-proneness</td>
<td>Selected set of metrics using information gain (IGS)</td>
</tr>
<tr>
<td>A8</td>
<td>Change-proneness</td>
<td>Selected set of metrics using univariate logistic regression analysis (LCS)</td>
</tr>
<tr>
<td>A9</td>
<td>Change-proneness</td>
<td>Extracted feature attributes using PCA</td>
</tr>
<tr>
<td>A10</td>
<td>Change-proneness</td>
<td>Reduced feature attributes using RSA</td>
</tr>
</tbody>
</table>
Effectiveness of Metrics

List of Metrics

Ten different set of source code metrics All metrics (AM), cohesion metrics (CHM), coupling metrics (CPM), size metrics (SM), inheritance metrics (IHM), selected set of metrics using gain ratio feature (GRS), selected set of metrics using information gain (IGS), selected set of metrics using univariate logistic regression analysis (LCS), extracted set of metrics using principal component analysis (PCA), selected set of metrics using rough set analysis (RSA)
We apply 4 different feature selection methods as the number of dimensions in our dataset is high.

Our objective is to eliminate some of the irrelevant and redundant original variables to increase the training speed and accuracy of the classifier.

We apply a filter approach for feature selection which precedes the classifier design and is also independent of the learning algorithm.

The four feature selection techniques that we use in our study are (1) Univariate Logistic Regression (2) Principal Component Analysis (3) Information Gain and Gain Ratio (4) Rough Set Analysis.
Table of Contents

1 Research Motivation and Aim
   - Research Motivation
   - Research Aim

2 Related work

3 Experimental Dataset
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4 Feature Selection Methods
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5 Research Method

6 Model Building Results

7 Conclusion

8 References

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Empirical Analysis on Effectiveness of Source Code Metrics for P
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Coeff.</th>
<th>p-val</th>
<th>Metrics</th>
<th>Coeff.</th>
<th>p-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA</td>
<td>0.044</td>
<td>0.028</td>
<td>NEWCO</td>
<td>-1.436</td>
<td>0</td>
</tr>
<tr>
<td>CAMC</td>
<td>-4.399</td>
<td>0</td>
<td>NEWICOM5</td>
<td>-2.434</td>
<td>0</td>
</tr>
<tr>
<td>CO</td>
<td>-0.089</td>
<td>0.246</td>
<td>OCC</td>
<td>-0.218</td>
<td>0.056</td>
</tr>
<tr>
<td>DCD</td>
<td>-0.694</td>
<td>0</td>
<td>PCC</td>
<td>0.015</td>
<td>0.886</td>
</tr>
<tr>
<td>DCI</td>
<td>-0.433</td>
<td>0</td>
<td>SNHD</td>
<td>-0.577</td>
<td>0</td>
</tr>
<tr>
<td>ICH</td>
<td>0.034</td>
<td>0</td>
<td>TCC</td>
<td>-0.728</td>
<td>0</td>
</tr>
<tr>
<td>LCC</td>
<td>-0.489</td>
<td>0</td>
<td>ACAIC</td>
<td>0.555</td>
<td>0.003</td>
</tr>
<tr>
<td>LCOM1</td>
<td>0.004</td>
<td>0</td>
<td>ACMIC</td>
<td>0.157</td>
<td>0.014</td>
</tr>
<tr>
<td>LCOM2</td>
<td>0.005</td>
<td>0</td>
<td>AMMIC</td>
<td>0.076</td>
<td>0</td>
</tr>
<tr>
<td>LCOM3</td>
<td>0.146</td>
<td>0</td>
<td>CBI</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LCOM4</td>
<td>0.155</td>
<td>0</td>
<td>CBO</td>
<td>0.128</td>
<td>0</td>
</tr>
<tr>
<td>LCOM5</td>
<td>1.268</td>
<td>0</td>
<td>DAC</td>
<td>0.21</td>
<td>0</td>
</tr>
<tr>
<td>NHD</td>
<td>0.192</td>
<td>0</td>
<td>DCAES</td>
<td>0.005</td>
<td>0.598</td>
</tr>
</tbody>
</table>

Univariate Logistic Regression (ULR)
Principal Component Analysis (PCA)
Gain Ratio (GR) and Information Gain (IG)
### Metrics, Coefficient, p-value

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Coeff.</th>
<th>p-val</th>
<th>Metrics</th>
<th>Coeff.</th>
<th>p-val</th>
<th>Metrics</th>
<th>Coeff.</th>
<th>p-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMEC</td>
<td>0.019</td>
<td>0.326</td>
<td>AID</td>
<td>0.362</td>
<td>0</td>
<td>NOP</td>
<td>0.729</td>
<td>0</td>
</tr>
<tr>
<td>ICP</td>
<td>0.021</td>
<td>0</td>
<td>CLD</td>
<td>0.139</td>
<td>0.004</td>
<td>SIX</td>
<td>2.22</td>
<td>0</td>
</tr>
<tr>
<td>IHICP</td>
<td>0.03</td>
<td>0</td>
<td>DIT</td>
<td>0.362</td>
<td>0</td>
<td>SP</td>
<td>0.134</td>
<td>0</td>
</tr>
<tr>
<td>MPC</td>
<td>0.046</td>
<td>0</td>
<td>DP</td>
<td>0.04</td>
<td>0</td>
<td>SPA</td>
<td>0.521</td>
<td>0</td>
</tr>
<tr>
<td>NIHICP</td>
<td>0.025</td>
<td>0</td>
<td>DPA</td>
<td>0.151</td>
<td>0</td>
<td>SPD</td>
<td>0.044</td>
<td>0.028</td>
</tr>
<tr>
<td>OCAEC</td>
<td>0.01</td>
<td>0.083</td>
<td>DPD</td>
<td>0.013</td>
<td>0.001</td>
<td>NA</td>
<td>0.051</td>
<td>0</td>
</tr>
<tr>
<td>OCAIC</td>
<td>0.218</td>
<td>0</td>
<td>NMA</td>
<td>0.067</td>
<td>0</td>
<td>NAIMP</td>
<td>0.111</td>
<td>0</td>
</tr>
<tr>
<td>OCMEEC</td>
<td>0.001</td>
<td>0.492</td>
<td>NMI</td>
<td>0.013</td>
<td>0</td>
<td>NM</td>
<td>0.021</td>
<td>0</td>
</tr>
<tr>
<td>OMCMIC</td>
<td>0.08</td>
<td>0</td>
<td>NMO</td>
<td>0.185</td>
<td>0</td>
<td>NMIMP</td>
<td>0.084</td>
<td>0</td>
</tr>
<tr>
<td>OMMECE</td>
<td>0.002</td>
<td>0.006</td>
<td>NOA</td>
<td>0.362</td>
<td>0</td>
<td>NUMPA</td>
<td>0.064</td>
<td>0</td>
</tr>
<tr>
<td>OMMID</td>
<td>0.055</td>
<td>0</td>
<td>NOC</td>
<td>0.015</td>
<td>0.203</td>
<td>STMTS</td>
<td>0.009</td>
<td>0</td>
</tr>
<tr>
<td>RFC</td>
<td>0.003</td>
<td>0</td>
<td>NOPD</td>
<td>0.013</td>
<td>0.022</td>
<td>SLOC</td>
<td>0.012</td>
<td>0</td>
</tr>
</tbody>
</table>

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Empirical Analysis on Effectiveness of Source Code Metrics for Predicting Change-Proneness
Univariate Logistic Regression (ULR)

Results and Observations

Univariate logistic regression helps in computing the percent or extent of variance (a predictor of statistical relationship between two variables) in the dependent variable explained by the independent variables.

The p-value of metrics OCC, PCC, DCAEC, DCMEC, OCAEC, OCMEC, NOC and NOPD is greater than the commonly used alpha threshold or level of 0.05 and hence they are not statistically significant predictors.

54 of the 62 metrics have a low p-value value ranging between 0 and 0.05 and hence they are useful predictors for our change-prone estimator.
Table of Contents

1 Research Motivation and Aim
   - Research Motivation
   - Research Aim

2 Related work

3 Experimental Dataset
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4 Feature Selection Methods
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5 Research Method

6 Model Building Results

7 Conclusion

8 References
### Principal Components and Correlated Metrics

<table>
<thead>
<tr>
<th>PC</th>
<th>Eigenvalue</th>
<th>% Variance</th>
<th>Cumulative</th>
<th>Correlated Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC1</td>
<td>7.48</td>
<td>12.076</td>
<td>12.076</td>
<td>ICII, LCOM3, LCOM4, NEWLCOM5, AMMIC, CBO, ICP, IHIJP, MPC, NIHIJP, OCMIC, OCM-MEC, OMMID, RFC, DPA, NMA, NMO, NA, NAIMP, NM, NMIMP, NUMPA, STMTS, SLCOC</td>
</tr>
<tr>
<td>PC2</td>
<td>6.47</td>
<td>10.448</td>
<td>22.523</td>
<td>AID, DIT, NMI, NOA, NOP, SIX</td>
</tr>
<tr>
<td>PC3</td>
<td>5.67</td>
<td>9.149</td>
<td>31.667</td>
<td>COA, DCD, OCC, CBI, DCAEC, CLD, DP, DPD, NDC, NOD, NOPD, SP</td>
</tr>
<tr>
<td>PC4</td>
<td>5.32</td>
<td>8.589</td>
<td>40.256</td>
<td>CO, DCI, LCC, NEWCO, PCC, TCC</td>
</tr>
<tr>
<td>PC5</td>
<td>4.01</td>
<td>6.477</td>
<td>46.734</td>
<td>LCOM1, LCOM2</td>
</tr>
<tr>
<td>PC6</td>
<td>3.89</td>
<td>6.274</td>
<td>53.007</td>
<td>DCMEC, OCAEC, OCMEC</td>
</tr>
<tr>
<td>PC7</td>
<td>2.61</td>
<td>4.207</td>
<td>57.215</td>
<td>CAMC</td>
</tr>
<tr>
<td>PC8</td>
<td>2.47</td>
<td>3.985</td>
<td>61.2</td>
<td>DAC, OCAIC</td>
</tr>
<tr>
<td>PC9</td>
<td>2.22</td>
<td>3.583</td>
<td>64.783</td>
<td>LCOM5</td>
</tr>
<tr>
<td>PC10</td>
<td>2.14</td>
<td>3.456</td>
<td>68.239</td>
<td>NHD, ACMIC</td>
</tr>
<tr>
<td>PC11</td>
<td>2.13</td>
<td>3.448</td>
<td>71.687</td>
<td>SNHD</td>
</tr>
<tr>
<td>PC12</td>
<td>2.12</td>
<td>3.428</td>
<td>75.115</td>
<td>ACAIC</td>
</tr>
<tr>
<td>PC13</td>
<td>1.9</td>
<td>3.209</td>
<td>78.324</td>
<td>SPA</td>
</tr>
<tr>
<td>PC14</td>
<td>1.85</td>
<td>3.150</td>
<td>80.004</td>
<td>SPD</td>
</tr>
</tbody>
</table>
Results and Observations

Table shows the eigenvalues of the 14 principal components in the decreasing order of eigenvalues and the proportion of variance (in terms of percentage variance) explained by the 14 principal components.

Results reveals the correlations between the 14 principal components and the 62 to source code metrics.

AID, DIT, NMI, NOA, NOP and SIX are most strongly correlated with the PC2 component. We infer from PCA procedure that AID, DIT, NMI, NOA, NOP and SIX vary together.
# Table of Contents

1. Research Motivation and Aim
   - Research Motivation
   - Research Aim

2. Related work

3. Experimental Dataset
   - Source Code Metrics
   - Correlation Analysis
   - Effectiveness of Metrics

4. Feature Selection Methods
   - Univariate Logistic Regression (ULR)
   - Principal Component Analysis (PCA)
   - Gain Ratio (GR) and Information Gain (IG)

5. Research Method

6. Model Building Results

7. Conclusion

8. References
### Feature Selection Experimental Results - I

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Gain-Ratio</th>
<th>Info-Gain</th>
<th>Metrics</th>
<th>Gain-Ratio</th>
<th>Info-Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA</td>
<td>0.001</td>
<td>0.004</td>
<td>NEWCO</td>
<td>0.027</td>
<td>0.042</td>
</tr>
<tr>
<td>CAMC</td>
<td>0.046</td>
<td>0.088</td>
<td>NEWLCOM5</td>
<td>0.04</td>
<td>0.08</td>
</tr>
<tr>
<td>CO</td>
<td>0.023</td>
<td>0.063</td>
<td>OCC</td>
<td>0.019</td>
<td>0.021</td>
</tr>
<tr>
<td>DCD</td>
<td>0.018</td>
<td>0.017</td>
<td>PCC</td>
<td>0.013</td>
<td>0.015</td>
</tr>
<tr>
<td>DCI</td>
<td>0.013</td>
<td>0.022</td>
<td>SNHD</td>
<td>0.029</td>
<td>0.039</td>
</tr>
<tr>
<td>ICH</td>
<td>0.042</td>
<td>0.081</td>
<td>TCC</td>
<td>0.019</td>
<td>0.018</td>
</tr>
<tr>
<td>LCC</td>
<td>0.015</td>
<td>0.014</td>
<td>ACAIC</td>
<td>0.01</td>
<td>0.002</td>
</tr>
<tr>
<td>LCOM1</td>
<td>0.05</td>
<td>0.094</td>
<td>ACMIC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LCOM2</td>
<td>0.048</td>
<td>0.089</td>
<td>AMMIEC</td>
<td>0.038</td>
<td>0.065</td>
</tr>
<tr>
<td>LCOM3</td>
<td>0.035</td>
<td>0.078</td>
<td>CBI</td>
<td>0.033</td>
<td>0.011</td>
</tr>
<tr>
<td>LCOM4</td>
<td>0.026</td>
<td>0.03</td>
<td>CBO</td>
<td>0.068</td>
<td>0.176</td>
</tr>
<tr>
<td>LCOM5</td>
<td>0.033</td>
<td>0.085</td>
<td>DAC</td>
<td>0.039</td>
<td>0.07</td>
</tr>
<tr>
<td>NHD</td>
<td>0.048</td>
<td>0.088</td>
<td>DCAEC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SP</td>
<td>0.022</td>
<td>0.017</td>
<td>SPA</td>
<td>0.022</td>
<td>0.014</td>
</tr>
<tr>
<td>NAIMP</td>
<td>0.032</td>
<td>0.062</td>
<td>NM</td>
<td>0.045</td>
<td>0.093</td>
</tr>
</tbody>
</table>
### Feature Selection Experimental Results - II

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Gain-Ratio</th>
<th>Info-Gain</th>
<th>Metrics</th>
<th>Gain-Ratio</th>
<th>Info-Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMSC</td>
<td>0</td>
<td>0</td>
<td>AID</td>
<td>0.022</td>
<td>0.035</td>
</tr>
<tr>
<td>ICP</td>
<td>0.084</td>
<td>0.167</td>
<td>CLD</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IHIICP</td>
<td>0.037</td>
<td>0.065</td>
<td>DIT</td>
<td>0.022</td>
<td>0.035</td>
</tr>
<tr>
<td>MPC</td>
<td>0.077</td>
<td>0.171</td>
<td>DP</td>
<td>0.027</td>
<td>0.042</td>
</tr>
<tr>
<td>NIHIICP</td>
<td>0.078</td>
<td>0.152</td>
<td>DPA</td>
<td>0.029</td>
<td>0.043</td>
</tr>
<tr>
<td>OCAEC</td>
<td>0.005</td>
<td>0.005</td>
<td>DPD</td>
<td>0.01</td>
<td>0.004</td>
</tr>
<tr>
<td>OCAIC</td>
<td>0.038</td>
<td>0.069</td>
<td>NMA</td>
<td>0.037</td>
<td>0.055</td>
</tr>
<tr>
<td>OCMSC</td>
<td>0</td>
<td>0</td>
<td>NMI</td>
<td>0.027</td>
<td>0.04</td>
</tr>
<tr>
<td>OCMIC</td>
<td>0.027</td>
<td>0.055</td>
<td>NMO</td>
<td>0.028</td>
<td>0.041</td>
</tr>
<tr>
<td>OMMSC</td>
<td>0</td>
<td>0</td>
<td>NOA</td>
<td>0.022</td>
<td>0.035</td>
</tr>
<tr>
<td>OMMID</td>
<td>0.072</td>
<td>0.156</td>
<td>NOC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RFC</td>
<td>0.06</td>
<td>0.141</td>
<td>NOPD</td>
<td>0.069</td>
<td>0.003</td>
</tr>
<tr>
<td>NOP</td>
<td>0.022</td>
<td>0.021</td>
<td>SIX</td>
<td>0.028</td>
<td>0.028</td>
</tr>
<tr>
<td>SPD</td>
<td>0.01</td>
<td>0.004</td>
<td>NA</td>
<td>0.041</td>
<td>0.089</td>
</tr>
<tr>
<td>NMIMP</td>
<td>0.044</td>
<td>0.086</td>
<td>NUMPA</td>
<td>0.039</td>
<td>0.088</td>
</tr>
</tbody>
</table>

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Empirical Analysis on Effectiveness of Source Code Metrics for Predicting Change Proneness
Results and Observations

We apply the procedure of selecting top $\lceil \log_2 n \rceil$ metrics out of $n$ metrics [61].

In our study $n = 62$ and hence we select the top 6 metrics. The 6 metrics selected using gain ratio are ICP, NIHICP, MPC, OMMID, STMTS, NOPD and the 6 metrics selected using information gain are CBO, MPC, ICP, OMMID, STMTS, NIHICP.

Table reveals the selected features represented using shaded gray cell.
Framework and Proposed Approach

Data Set → Normalization of the dataset → Partition of dataset → Classification Techniques with 5-fold cross validation → Performance Evaluation → Model comparison → Validation of developer models

- Framework and Proposed Approach

Lov Kumar, Santanu Kumar Rath, Ashish Sureka
We apply Eight different learning algorithms:

1. Logistic Regression (LOGR)
2. Naive Bayes Classifier (NBC)
3. Extreme Learning Machine (ELM) with linear (LIN), polynomial (PLY) and Radial Basis Function (RBF) kernels
4. Support Vector Machine (SVM) with linear (LIN), RBF and Sigmoid kernel (SIG)
5. Two ensemble techniques such as Best-in-Training (BTE) and Majority Voting (MV)
Best Training Ensemble (BTE)

Algorithm 1 Best Training Ensemble (BTE) Method

1: Select Data with N Number of features.
2: Select M number of classification models.
3: Select K for K-fold cross validation.
4: for each \( k \in K \) fold do
5: \hspace{1em} for each \( m \in M \) model do
6: \hspace{2em} Train model \( m \) on the training data of k-fold.
7: \hspace{2em} Apply model \( m \) on the training data of k-fold
8: \hspace{2em} Compute training performance of model \( m \) (\( P_m \)) based on certain performance parameter
9: \hspace{2em} Store the value of \( P_m \)
10: \hspace{1em} end for
11: Select best model \( M_b \in M \) model based on performance \( P_m \)
12: for each \( n \in N_{test} \) number of test data for fold \( k \) do
13: \hspace{2em} \( E_{out} \) = Result of model \( M_b \) on testing data \( n \)
14: \hspace{2em} end for
15: end for
Majority Voting Ensemble (MVE) Method

Algorithm 2 Majority Voting Ensemble (MVE) Method

1: Select Data with N Number of features.
2: Select M number of classification models.
3: Select K for K-fold cross validation.
4: for each \( k \in K \) fold do
5: \hspace{1em} for each \( m \in M \) model do
6: \hspace{2em} Apply model \( m \) on the training data of k-fold
7: \hspace{2em} Compute the output of trained model \( m \) on testing data of k-fold
8: \hspace{2em} Store the value of output for testing data of the trained model
9: \hspace{1em} end for
10: for each \( n \in N_{test} \) number of test data for fold \( k \) do
11: \hspace{1em} Count the number of models predicting a category
12: \hspace{1em} \( E_{out} \) = the category which has the maximum count on testing data \( n \)
13: \hspace{1em} end for
14: end for
We apply 8 different classification algorithms and 2 different ensemble techniques resulting in 10 different predictive model building approaches.

We evaluate the performance of these models using accuracy and AUC metrics.

Predicting change-proneness of classes is a binary classification problem and both accuracy and AUC are common evaluation metrics for such problems.
Evaluation Results for Ten Sub-Set of Metrics - Accuracy

![Graph showing accuracy results for different metrics.]

Lov Kumar, Santanu Kumar Rath, Ashish Sureka

Empirical Analysis on Effectiveness of Source Code Metrics for P
Evaluation Results for Ten Sub-Set of Metrics - AUC
Evaluation Results for Ten Classification Methods - Accuracy

![Box plot showing accuracy for ten classification methods](image-url)
Evaluation Results for Ten Classification Methods - AUC
Observations and Results

- In most of the cases, the model developed by considering selected set of metrics using feature selection techniques as input obtained better performance i.e., high values of accuracy and AUC for predicting change-proneness as compared to a model developed using all metrics.

- Extreme Learning Machine with polynomial kernel function (ELM-PLY) yields better results when compared to other classification.
Observations and Results

- Majority Voting (MV) ensemble method outperformed as compared to all other classifier except ELM-PLY kernel.
- Among different kernel function, polynomial kernel in ELM and RBF kernel in SVM yields better results compared to other kernel functions.
- Model developed using coupling metrics have high median value of performance parameters as compare to other three set of source code metrics. This results shows that coupling metrics have higher predictive ability as compared to size metrics, cohesion metrics and inheritance metrics.
Model developed using selected set of source code metrics using rough set analysis (RSA) have high median value of performance parameters as compare to other.

There exists a small subset of source code software metrics out of total available source code software metrics which are able to predict change-proneness with higher accuracy and reduced value of misclassified errors.
Summary and Takeaways

- Coupling metrics have higher predictive ability as compared to size metrics, cohesion metrics and inheritance metrics.
- From experimental results, it is observed that, there exists a small subset of source code software metrics out of total available source code software metrics which are able to predict change-proneness with higher accuracy and reduced value of misclassified errors.
Summary and Takeaways

- We observed that the model developed using RSA yields better result compared to other approaches.
- We observed that model developed using Extreme Learning Machine with polynomial kernel function (ELM-PLY) yields better result as compare to other classification techniques, we also observed that Majority Voting (MV) ensemble method outperformed as compared to all other classifier except ELM-PLY kernel.
- From experiments, it is observed that the performance of the feature selection techniques is varied with the difference classification methods used.
References I

Golnoush Abaei, Ali Selamat, and Hamido Fujita.
An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction.
*Knowledge-Based Systems.*

F. B. E Aberu.
The mood metrics set.

F. B. E. Abreu and R. Carapuca.
Object-Oriented software engineering: Measuring and controlling the development process.

Hojjat Adeli and Shih-Lin Hung.
References


Briand L.C Arisholm E and Johannessen E.B.
A systematic and comprehensive investigation of methods to build and evaluate fault prediction models.

J Scott Armstrong and Fred Collopy.
Error measures for generalizing about forecasting methods: Empirical comparisons.

Rajendra K. Bandi, Vijay K. Vaishnavi, and Daniel E Turk.
Predicting maintenance performance using object-oriented design complexity metrics.

Rajiv D Banker, Srikant M Datar, Chris F Kemerer, and Dani Zweig.
Software complexity and maintenance costs.


Victor R Basili, Lionel C Briand, and Walcélio L Melo. How reuse influences productivity in object-oriented systems. *Communications of the ACM.*
Dilek Baski and Sanjay Misra. 
Metrics suite for maintainability of extensible markup language web services. 

R Battiti. 
First and second-order methods for learning between steepest descent and newton’s method. 

Aaron B Binkley and Stephen R Schach. 
Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures. 

Barry W. Boehm. 
Software engineering economics. 
References VI


C.J. Burgess and M. Lefley.  
Can genetic programming improve software effort estimation.  

Rachel Burrows, Fabiano C Ferrari, Otavio AL Lemos, Alessandro Garcia, and Francois Taiani.  
The impact of coupling on the fault-proneness of aspect-oriented programs: an empirical study.  

Jones C.  
Software quality in 2010: a survey of the state of the art.  
*In Founder and Chief Scientist Emeritus*, 2010.
References IX

Massimo Carbone and Giuseppe Santucci.
Fast&&serious: a uml based metric for effort estimation.

Michelle Cartwright and Martin Shepperd.
An empirical investigation of an object-oriented software system.

Jie-Cherng Chen and Sun-Jen Huang.
An empirical analysis of the impact of software development problem factors on software maintainability.

Yuehui Chen, Ajith Abraham, and Bo Yang.
Feature selection and classification using flexible neural tree.
References


- Shyam R Chidamber and Chris F Kemerer. *Towards a metrics suite for Object-Oriented design*, volume 26. ACM.

Don Coleman, Bruce Lowther, and Paul Oman.
The application of software maintainability models in industrial software systems.


José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra.
Analyzing the evolution of web services using fine-grained changes.

José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra.
Predicting web service maintainability via object-oriented metrics: a statistics-based approach.
References XII

Ana Erika Camargo Cruz and Koichiro Ochimizu.
Towards logistic regression models for predicting fault-prone code across software projects.

Melis Dagpinar and Jens H Jahnke.
Predicting maintainability with object-oriented metrics-an empirical comparison.

Leandro N De Castro and Fernando J Von Zuben.
Learning and optimization using the clonal selection principle.
Shyamala Doraisamy, Shahram Golzari, Noris Mohd, Md Nasir Sulaiman, and Nur Izura Udzir.
A study on feature selection and classification techniques for automatic genre classification of traditional malay music.

Didier Dubois and Henri Prade.
Fuzzy real algebra: some results.

R. Slowinski (Ed).

Khaled El Emam, Saïda Benlarbi, Nishith Goel, and Shesh N. Rai.
The confounding effect of class size on the validity of object-oriented metrics.


References XV

Emile Fiesler.
Neural network classification and formalization.

Marios Fokaefs, Rimon Mikhail, Nikolaos Tsantalis, Eleni Stroulia, and Alex Lau.
An empirical study on web service evolution.

George Forman.
An extensive empirical study of feature selection metrics for text classification.

Jerome H Friedman.
Multivariate adaptive regression splines.


Rinkaj Goyal, Pravin Chandra, and Yogesh Singh. Suitability of knn regression in the development of interaction based software fault prediction models. *IERI Procedia*.

References XVIII


References XIX


Allen E.B Hudepohl J.P Hochman R, Khoshgoftar T.M.  
Evolutionary neural networks: a robust approach to software reliability problems.  

D Huang and Tommy WS Chow.  
Effective feature selection scheme using mutual information.  

Ross Huitt and Norman Wilde.  
Maintenance support for object-oriented programs.  

Rob J Hyndman and Anne B Koehler.  
Another look at measures of forecast accuracy.  
A. Idri, A. Abran, and S. Mbarki.  
An experiment on the design of radial basis function neural networks for software cost estimation.  

Y Kumar Jain and Santosh Kumar Bhandare.  
Min max normalization based data perturbation method for privacy protection.  

Cukic B Jiang Y and Ma Y.  
Techniques for evaluating fault prediction models.  
References


Lov Kumar, Santanu Kumar Rath, Ashish Sureka. Empirical Analysis on Effectiveness of Source Code Metrics for P
References XXIII


References XXV


R. Kohavi.
Relation between software metrics and maintainability.

Ron Kohavi and George H John.
Wrappers for feature subset selection.

P. Devanbu L. Briand and W. Melo.
An investigation into coupling measures for C++.
References

Al Lake and Curtis Cook.
Use of factor analysis to develop oop software complexity metrics.

YS Lee, BS Liang, SF Wu, and FJ Wang.
Measuring the coupling and cohesion of an object-oriented program based on information flow.

K. Levenberg.
A method for the solution of certain non-linear problems in least squares.


References XXIX


References

Ruchika Malhotra and Yogesh Singh.
On the applicability of machine learning techniques for object oriented software fault prediction.
*Software Engineering: An International Journal.*

Andrian Marcus and Denys Poshyvanyk.
The conceptual cohesion of classes.

D. W. Marquardt.
An algorithm for the least-squares estimation of nonlinear parameters.

R. Martin.
Object-oriented design quality metrics an analysis of dependencies.
References XXXI

Cristian Mateos, Marco Crasso, Alejandro Zunino, and Jos? Luis Ordiales Coscia.
Detecting wsdl bad practices in code–first web services.

Thomas J. McCabe.
A complexity measure.

Andrew McCallum and Kamal Nigam.
A comparison of event models for naive bayes text classification.

Warren McCulloch and Walter Pitts.
A logical calculus of ideas immanent in nervous activity.
References XXXII

W. Melo and F. B. E. Abreu.
Evaluating the impact of Object-Oriented design on software quality.

Rui Mendes, Paulo Cortez, Miguel Rocha, and José Neves.
Particle swarms for feedforward neural network training.
*learning*, 6(1), 2002.

Tim Menzies, Bora Caglayan, Zhimin He, Ekrem Kocaguneli, Joe Krall, Fayola Peters, and Burak Turhan.
The promise repository of empirical software engineering data, June 2012.

Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum.
Selecting best practices for effort estimation.
Bharavi Mishra, KK Shukla, et al.
Defect prediction for object oriented software using support vector based fuzzy classification model.
*International Journal of Computer Applications.*

Subhas Chandra Misra.
Modeling design/coding factors that drive maintainability of software systems.

Specification and detection of soa antipatterns.

J Moody and J Darken C.
Fast learning in networks of locally-tunes processing units.


Hector M Olague, Letha H Etzkorn, Sampson Gholston, and Stephen Quattlebaum.
Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes.
*Software Engineering, IEEE Transactions on.*

Hector M Olague, Letha H Etzkorn, Sampson Gholston, and Stephen Quattlebaum.
Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes.

Paul Oman and Jack Hagemeister.
Construction and testing of polynomials predicting software maintainability.


References XXXVII


Lov Kumar, Santanu Kumar Rath, Ashish Sureka. Empirical Analysis on Effectiveness of Source Code Metrics for P

Raed Shatnawi and Wei Li. The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process. Journal of systems and software.


K. Srinivasan and D. Fisher.
Machine learning approaches to estimating software development effort. 

Ramanath Subramanyam and Mayuram S. Krishnan.
Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. 

Mende T and Koschke R.

Mende T and Koschke R.
References XLI

Empirical validation of Object-Oriented metrics on open source software for fault prediction.  

Mei-Huei Tang, Ming-Hung Kao, and Mei-Hwa Chen.  
An empirical study on object-oriented metrics.  

D. P. Tegarden, S. D. Sheetz, and D. E. Monarchi.  
A software complexity model of Object-Oriented systems.  

K Thangavel, Qiang Shen, and A Pethalakshmi.  
Application of clustering for feature selection based on rough set theory approach.  
References XLII


References XLIII

- **G. Witting and G. Finnie.**
  Estimating software development effort with connectionist models.

- **W. Li and S. Henry.**
  Object-oriented metrics that predicts maintainability.

- **Fangjun Wu.**
  Empirical validation of object-oriented metrics on nasa for fault prediction.

- **Yijun Yu, Jianguo Lu, Juan Fernandez-Ramil, and Phil Yuan.**
  Comparing web services with other software components.
Jidong Zhao, Ke Lu, and Xiaofei He. 
Locality sensitive semi-supervised feature selection.  

Jialin Zhou, Zhengcheng Duan, Yong Li, Jianchun Deng, and Daoyuan Yu.  
Pso-based neural network optimization and its utilization in a boring machine. 

Yuming Zhou and Hareton Leung. 
Empirical analysis of object-oriented design metrics for predicting high and low severity faults. 

Yuming Zhou and Hareton Leung. 
Predicting object-oriented software maintainability using multivariate adaptive regression splines. 
References
