Aging Related Bug Prediction using Extreme Learning Machines
Lov Kumar (lovkumar505@gmail.com) and Ashish Sureka (ashish.sureka@ieee.org)
Thapar University and Ashoka University

Abstract:
1) Aging-Related Bugs (ARBs): Error-conditions associated with software aging such as memory leakage or unreleased files and locks.
2) Approach:
   i. Source code metrics and machine learning techniques used to predict aging related bugs.
   ii. Five different feature selection techniques are considered for dimensionality reduction.
   iii. SMOTE method to counter the effect of class imbalance.
   iv. Extreme Learning Machines (ELM) with three different kernels are considered for model building.

Introduction:
1) Hard to uncover aging related bugs during system testing.
2) Recent research shows that static source code metrics can be used as predictors for aging related bugs.
3) Challenges in building predictive models
   i. Imbalanced Data.
   ii. High-Dimensional Data.

Research questions:
RQ1: Is there a statistically significant difference in the prediction performance of five different feature ranking techniques? Which feature ranking technique(s) yields the best result?
RQ2: Is there any statistically significant difference in the prediction performance of the predictive model after applying SMOTE method to address the class imbalance problem?
RQ3: What is the relative performance of the ELM with three different types of kernels?

Experimental Dataset:
Publicly available dataset, tera PROMISE.

<table>
<thead>
<tr>
<th>Name</th>
<th># Records</th>
<th>Majority Class</th>
<th>Minority Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux driver net</td>
<td>2292</td>
<td>2283</td>
<td>9</td>
</tr>
<tr>
<td>Linux driver sxe</td>
<td>962</td>
<td>958</td>
<td>4</td>
</tr>
<tr>
<td>Linux ext3</td>
<td>29</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>Linux ipv4</td>
<td>117</td>
<td>115</td>
<td>2</td>
</tr>
</tbody>
</table>

Experimental Results:

metrics sets: Boxplot Showing the Degree of Dispersion in the AUC Value without and with SMOTE

Kernel: Boxplot Showing the Degree of Dispersion in the AUC Value without and with SMOTE

Descriptive Statistics of the Relative Performance of SMOTE and Without SMOTE in terms of AUC

<table>
<thead>
<tr>
<th>AUC</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Std_DEV</th>
<th>Q1</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without SMOTE</td>
<td>0.080</td>
<td>1.000</td>
<td>0.51</td>
<td>0.169</td>
<td>0.159</td>
<td>0.500</td>
<td>0.510</td>
</tr>
<tr>
<td>With SMOTE</td>
<td>0.200</td>
<td>1.000</td>
<td>0.51</td>
<td>0.169</td>
<td>0.159</td>
<td>0.500</td>
<td>0.510</td>
</tr>
</tbody>
</table>

Conclusion:
1) There are several metrics which can be extracted from the software system, not all are equally important as several metrics are redundant and irrelevant.
2) Feature ranking techniques improve the performance of the predictive model in comparison to all metrics but there is no significant difference in their performance.
3) SMOTE improves the performance for ELM linear kernel but not polynomial or RBF kernel.
4) Performance of 3 different ELM kernel functions are significantly different according.

References: