Meta-heuristic Feature selection for Software Defect Prediction

Yuji Jiang

Software Defect

Definition

Software Defect means any documented occurrence of an instance where the software does not perform according to its published specifications.

Consequence

- Affect software’s basic functionality
- May have ripple effects
- Loss of money, time, and resource for both developers and investors

Software Defect

Common Methods to deal with defect

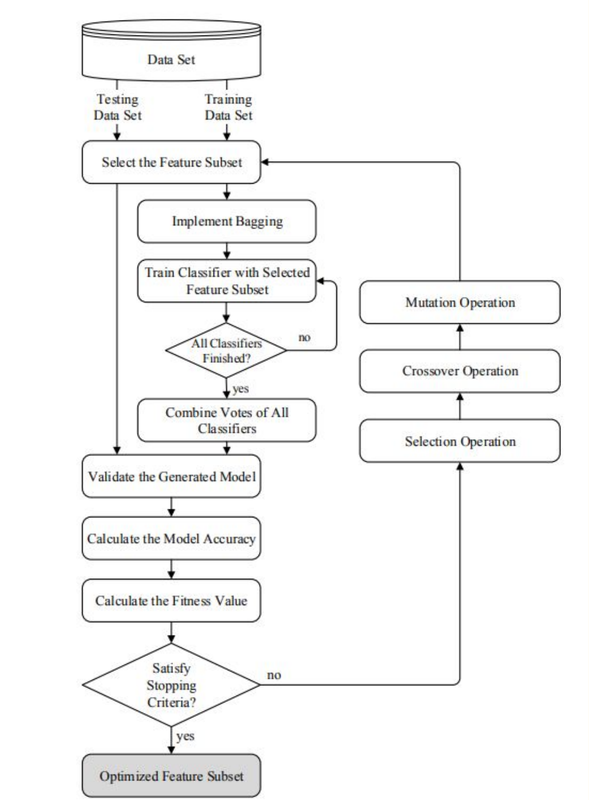
- **Code reviews**
  - Time and Resource constraints
- **Static code analysis (AST)**
  - Too many false positives
  - Rely on specific SW language
  - Expensive
- **Unit testing**
  - Takes time
  - Difficult to write for legacy codes

By using Machine learning to predict Software Defect

- **Reduce Cost**
- **Improve Accuracy**
- **Does not depend on Software Language or specific system environment**

Paper: Genetic Feature Selection for Software Defect Prediction

- **Decreases over-fitting**
  - Fewer redundant data means fewer chances of making decisions based on noise.
- **Improves Accuracy**
  - Less misleading data means better modeling accuracy.
- **Reduces Training Time**
  - Less data means quicker algorithms.



Notable Procedures

In the Paper:

- Bagging
- Genetic Algorithm

Extra:

- Hyperparameter tuning
- Boosting vs Bagging
- Island-based GA
- Physical Annealing
- Hill Climbing

Characteristics of Procedure

NASA MDP Dataset

- Serious class imbalance issue
  - Much more Negative than positive
- Extracted from different coding language
  - C, C++, JAVA, etc.

Fitness Function

- The paper provided a abstract equation but did not define the equation well.
- Did not define terms like “Feature Value” as well as “Feature Cost”

$$fitness = W_A \times A + W_F \times \left(P + \left(\sum_{i=1}^{n_f} C_i \times F_i \right) \right)^{-1}$$

Result

From the Paper

Table 2. AUC of 10 Classifiers on 9 Data Sets (without GA and Bagging)

Classifiers		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
Statistical Classifier	LR	0.763	0.801	0.713	0.766	0.726	0.852	0.849	0.81	0.894
	LDA	0.471	0.536	0.447	0.503	0.58	0.454	0.577	0.524	0.61
	NB	0.734	0.786	0.67	0.739	0.732	0.781	0.811	0.756	0.838
Nearest Neighbor	k-NN	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	K*	0.6	0.678	0.562	0.585	0.63	0.652	0.754	0.697	0.76
Neural Network	BP	0.713	0.791	0.647	0.71	0.625	0.784	0.918	0.79	0.883
Support Vector Machine	SVM	0.753	0.752	0.642	0.761	0.714	0.79	0.534	0.75	0.899
Decision Tree	C4.5	0.565	0.515	0.497	0.455	0.543	0.601	0.493	0.715	0.723
	CART	0.604	0.648	0.637	0.482	0.656	0.574	0.491	0.68	0.623
	RF	0.573	0.485	0.477	0.525	0.74	0.618	0.649	0.678	0.2

Table 3. AUC of 10 Classifiers on 9 Data Sets (with GA and Bagging)

Classifiers		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
Statistical Classifier	LR	0.753	0.795	0.691	0.761	**0.742**	0.852	0.822	0.813	**0.901**
	LDA	**0.592**	**0.627**	**0.635**	**0.64**	**0.674**	**0.637**	**0.607**	**0.635**	**0.715**
	NB	0.702	**0.79**	**0.677**	0.739	0.724	**0.799**	0.805	**0.78**	**0.861**
Nearest Neighbor	k-NN	**0.666**	**0.689**	**0.67**	**0.783**	**0.656**	**0.734**	**0.554**	**0.649**	**0.732**
	K*	**0.71**	**0.822**	0.503	**0.718**	**0.68**	**0.876**	**0.877**	**0.816**	**0.893**
Neural Network	BP	**0.744**	**0.797**	**0.707**	**0.835**	**0.689**	**0.829**	0.905	**0.799**	**0.921**
Support Vector Machine	SVM	0.667	**0.767**	0.572	**0.747**	**0.659**	0.774	0.139	0.476	**0.879**
Decision Tree	C4.5	**0.64**	**0.618**	**0.658**	**0.732**	**0.695**	**0.758**	**0.642**	**0.73**	**0.844**
	CART	**0.674**	**0.818**	**0.754**	**0.709**	**0.703**	**0.819**	**0.832**	**0.842**	**0.9**
	RF	**0.706**	**0.584**	**0.605**	0.483	0.735	**0.696**	**0.901**	**0.734**	**0.601**

Results from Project

Without GA

AUC W/O GA	CM1	KC1	KC3
LR	0.592	0.649	0.517
LDA	0.733	0.681	0.567
NB	0.732	0.652	0.6
K-NN	0.5	0.612	0.5
C4.5	0.674	0.622	0.617
CART	0.532	0.666	0.617

GA & Bagging

AUC with GA&Bagging	CM1	KC1	KC3
LR	**0.8**	**0.666**	**0.6**
LDA	**0.75**	**0.720**	**0.65**
NB	**0.75**	0.634	**0.683**
K-NN	**0.65**	**0.665**	**0.6**
C4.5	**0.694**	**0.650**	**0.7**
CART	**0.7**	0.650	**0.75**

GA & Boosting

AUC with GA&Boosting	CM1	KC1	KC3
LR	**0.75**	**0.672**	**0.6**
LDA	**0.8**	**0.715**	**0.65**
NB	**0.75**	0.634	**0.7**
K-NN	**0.65**	**0.678**	**0.6**
C4.5	**0.7**	**0.672**	**0.7**
CART	**0.75**	**0.671**	**0.75**

Further Study

CM1 CART Comparison

CM1-DT	Regular	GA	GA+HT	IGA	PA	HC
Accuracy	0.763	0.908	0.908	0.918	0.889	0.879
F1	0.189	0.535	0.535	0.621	0.497	0.357
Precision	0.186	1.0	1.0	0.947	0.783	0.783
Recall	0.2	0.367	0.367	0.467	0.4	0.233
Auc	0.530	0.683	0.683	0.731	0.686	0.611
Time	0.005	36.43	45.23	88.3	6.90	1.84

Conclusion

- Overall, feature selection using Meta-heuristic does help improving the accuracy of the training model as well as reducing complexity.
- Island-based Genetic Algorithm gives the best result, while also being the most computationally expensive.
- Bagging and Boosting achieves similar results, boosting performs slightly better due to perhaps how it aims at improving bias.
- Physical Annealing provides decent results, but is inconsistent over different runs.