Spectral and neural gas clustering techniques for software defect prediction: performance evaluation with different feature selection methods

Gianluca Bertaccini

Master's degree in Statistical Sciences (Data Science)
University of Bologna

gianluca.bertaccini@studio.unibo.it

March 29, 2022

Software defect prediction

- Predicting software defects is essential for software developers: if they are able to **identify in advance defective instances of code,** they can spend their time and resources fixing them rather than the clean ones.
- Supervised and unsupervised models can correctly identify software defects by analyzing datasets of **software metrics,** directly computed from the source code.
- In general, software defect prediction makes the process of software development more efficient.

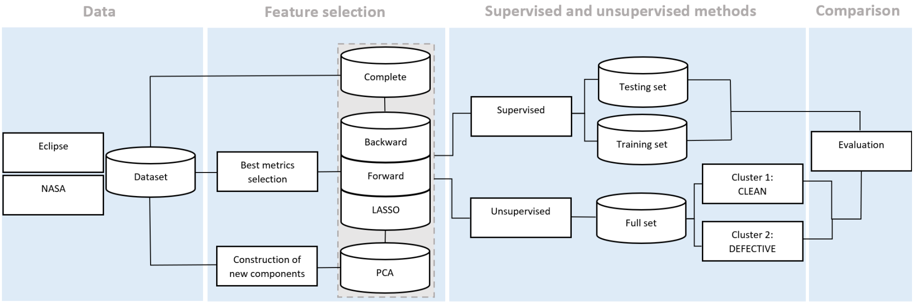
Research questions

These are the research questions of software defect prediction that are considered in this project:

- **RQ1.** Do current unsupervised methods perform better or worse than the supervised ones?
- **RQ2.** Which feature selection method is the best for supervised methods?
- **RQ3.** Which feature selection method is the best for unsupervised methods?

Methodology

Methodology followed to answer the research questions:



Data

To each project corresponds a dataset. There are in total:

- 5 Eclipse datasets
- 12 NASA datasets

Eclipse projects	Description
Eclipse JDT Core	IDE for software developers
Equinox Framework	for development of modular software programs
Apache **Lucene**	indexing and search features
Mylyn	task management tools
Eclipse **PDE** UI	plug-in development environment to develop Eclipse features
NASA projects	Description
CM1	software for a spacecraft instrument
JM1	simulate ground predictions
KC1, KC3	storage management software
MC1, MC2, MW2	no information available
PC1, PC2, PC3, PC4, PC5	software satellite

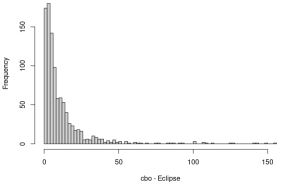
Data: Eclipse

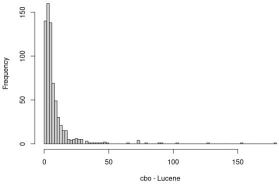
Eclipse datasets contain 18 numeric variables:

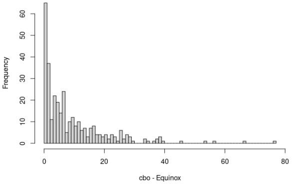
- **6 CK** software metrics: they describe the main characteristics of a class (or different instance) of code.
  - **cbo:** coupling between objects
  - **dit:** depth of inheritance tree
  - **lcom:** lack of cohesion methods
  - **noc:** number of children
  - **rfc:** response for a class
  - **wmc:** weighted method count
- **11 object-oriented** software metrics. They describe further characteristics of classes of code, such as number of lines, number of methods or number of attributes.
- **1 binary response** variable, labeling the instance as defective or not (1 or 0).

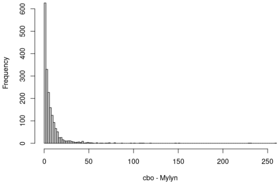
Data: Eclipse, histograms for cbo variable

The data is not normally distributed. For example, the histograms of cbo variable are highly skewed.



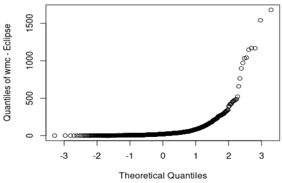


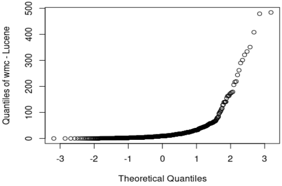


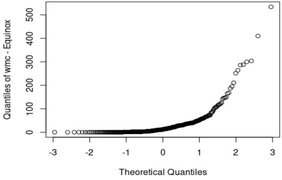


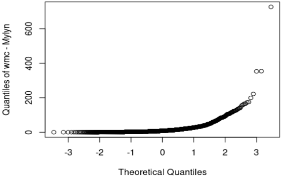
Data: Eclipse, quantile plots for wmc variable

The data is not normally distributed. For example, the quantile plots of the wmc variable show a strong deviation from normality.





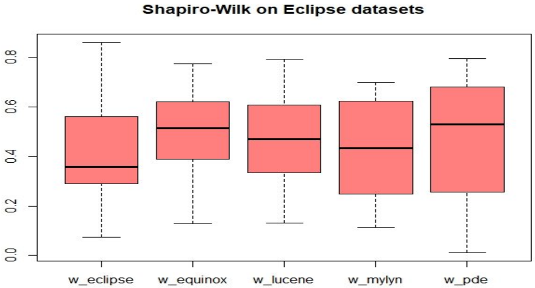




Data: Eclipse, Shapiro-Wilk tests

The average Shapiro-Wilk test for every dataset is in the interval between 0.4 and 0.6, which means the variables are not normally distributed. For example, these are the variables lcom and nopa:

Variables	W_Eclipse	W_Equinox	W_Lucene	W_Mylyn	W_PDE
lcom	0.07	0.37	0.13	0.21	0.37
nopa	0.14	0.13	0.47	0.11	0.01

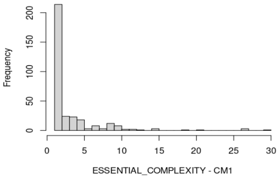


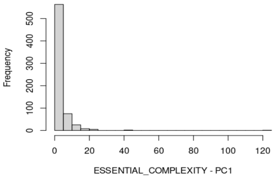
Data: NASA

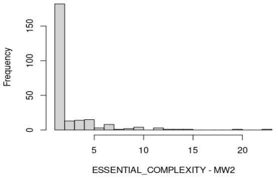
NASA datasets contain different number of variables, from 22 to 40 variables in total (depending on the dataset). These variables are all numeric and belong to different categories:

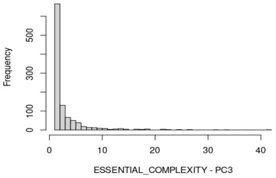
- **Lines of code metrics:** counting the number of lines of code, comments, percentages.
- **McCabe's complexity metrics:** metrics computed starting from the McCabe's Cyclomatic Complexity (number of conditional branches in the control-flow graph of an instance of code), described as $\mathbf{M = E - N + 2P}$ where E is the number of edges, N is the number of nodes and P is the number of connected components.
- **Halstead complexity metrics:** different complexity metrics that are computed starting from the number of operands and operators.
- **Other complexity metrics:** complexity metrics that are not based directly on McCabe's or Halstead's complexity.
- **Density metrics:** complexity metrics with respect to the number of total paths in an instance of code.
- **1 binary response** variable, labeling the instance as defective or not.

The data is not normally distributed. For example, the histograms of ESSENTIAL_COMPLEXITY variable are highly skewed.



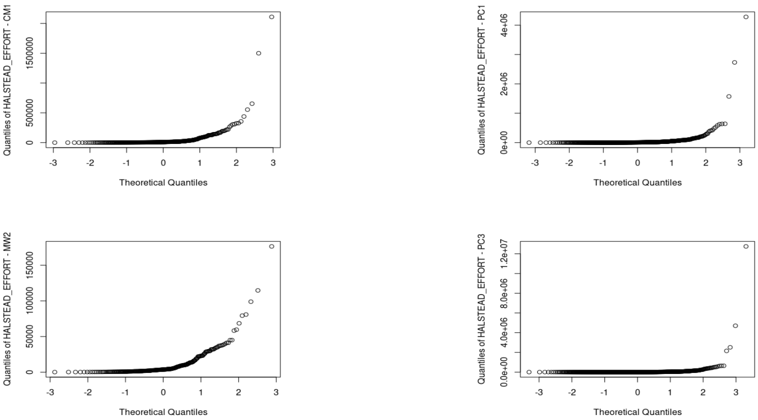






Data: NASA, quantile plots for HALSTEAD_EFFORT

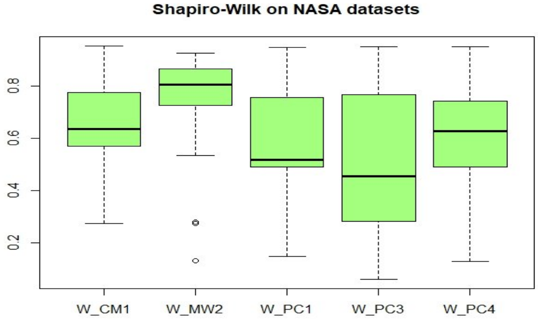
The data is not normally distributed. For example, the quantile plots of the HALSTEAD_EFFORT variable show a strong deviation from normality.



Data: NASA, Shapiro-Wilk tests

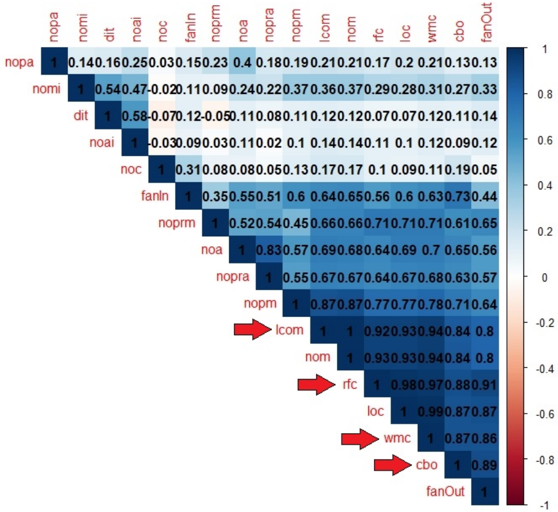
The average Shapiro-Wilk test for NASA datasets is a bit higher than for the Eclipse ones. It reaches up to 0.8. Most variables are still not normally distributed, except for:

Variables	W_Eclipse	W_Equinox	W_Lucene	W_Mylyn	W_PDE
CYCLOMATIC_DENSITY	0.91	0.93	0.95	0.95	0.88
NORMALIZED_CYCLOMATIC_COMPLEXITY	0.86	0.92	0.94	0.93	0.85
NUM_UNIQUE_OPERATORS	0.86	0.90	0.82	0.90	0.95
PERCENT_COMMENTS	0.95	0.91	0.82	0.83	0.85



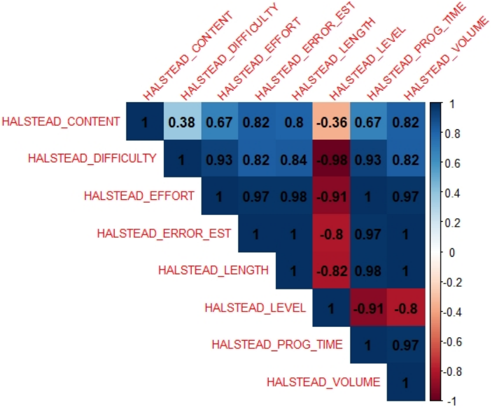
Correlations: Eclipse

Eclipse datasets have high correlations for the **CK** metrics (dit, noc, **rfc, wmc, cbo, lcom).**



Correlations: NASA

NASA datasets have high correlations for the **Halstead** metrics. This can influence negatively the performance of machine learning methods.

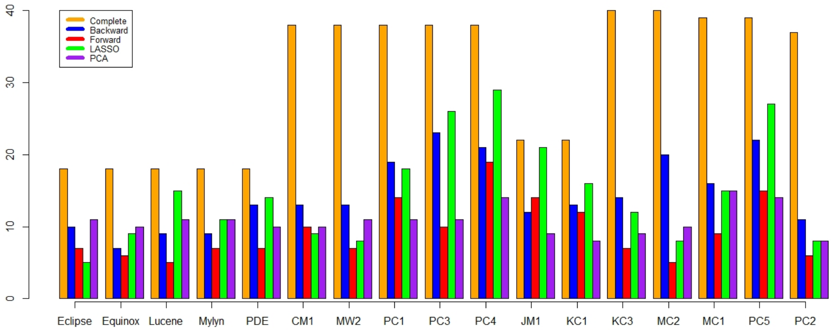


Feature selection

- To reduce the correlations between software metrics, different approaches of feature selection (or feature construction) are tested with each machine learning method.
- Choosing a smaller subset of features can reduce correlations and improve the algorithms performance.
- The approaches are:
  - **Backward** Elimination
  - **Forward** Selection
  - Least Absolute Shrinkage and Selection Operator (**LASSO**)
  - Principal Components Analysis (**PCA**)

Feature selection

Number of features selected by different feature selection approaches.



Methods

Supervised machine learning methods achieve good performance, but **unsupervised methods** are being introduced because many software defect datasets often:

- lack historical data
- have unlabeled or incomplete data

Hence, two clustering techniques are studied in this project and compared to common supervised classifiers:

- **Naive Bayes**
- **Random Forest**
- **Neural Gas Clustering**
- **Spectral Clustering**

Performance of methods

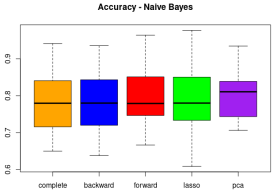
Each supervised and unsupervised method, along with the four different feature selection approaches, is compared to the others by the following evaluation measures:

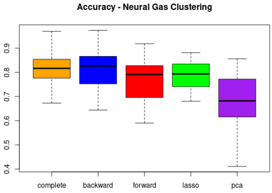
- **Accuracy** $= \frac{TP+TN}{TP+TN+FP+FN}$.
- Precision $= \frac{TP}{TP+FP}$.
- Recall $= \frac{TP}{TP+FN}$.
- Area Under the Curve (**AUC**): this curve is made by plotting the recall (or TPR, true positive rate) against the FPR (false positive rate), which can be computed as $1 - \mathit{specificity}$.

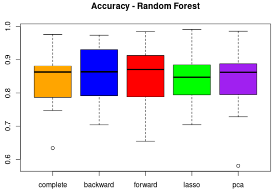
The comparison is focused in particular on Accuracy and AUC measures, the most common in software defect prediction.

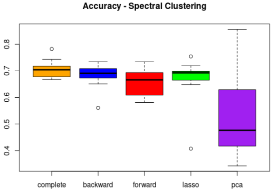
Performance of methods: accuracy

- According to accuracy, **Random Forest** achieves the **best performance** for all feature selection approaches.
- For **unsupervised** methods, **PCA** gives results **significantly worse** than the other feature selection approaches.



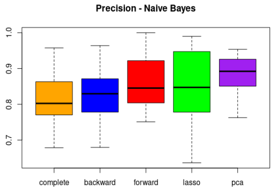


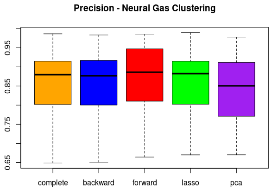


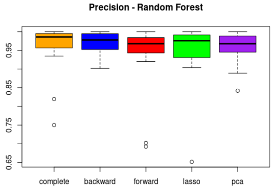


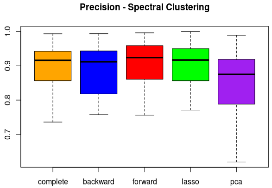
Performance of methods: precision

- **Precision** values are high (on average **above 0.8**) for all methods, and **Random Forest** achieves again the best performance.
- There is not a significant difference between different feature selection approaches. **PCA** performs as well as the others.



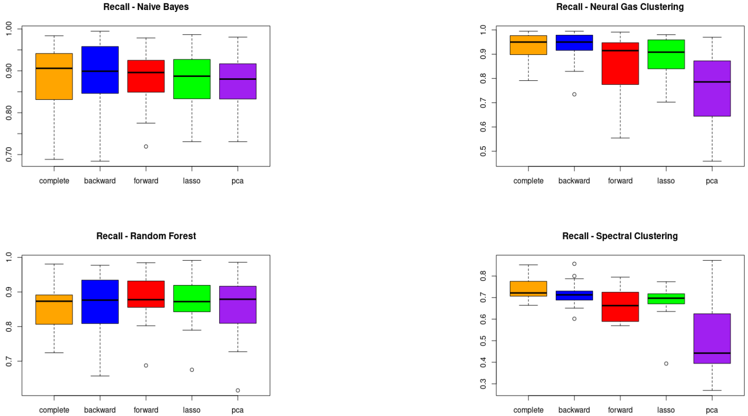






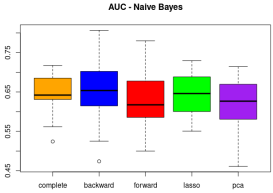
Performance of methods: recall

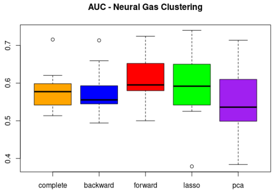
- The **recall** values are very high **(around 0.9)** for all the methods except spectral clustering. This means most non-defective classes are correctly identified.
- As for accuracy, **PCA negatively influences** the performance of **unsupervised** methods.

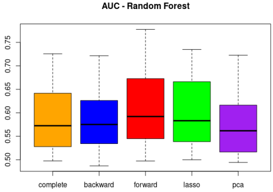


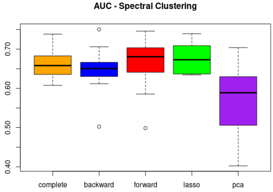
Performance of methods: AUC

- AUC values are lower than other evaluation measures, with **Spectral Clustering** having on average the best performance **(around 0.65).**
- For unsupervised methods, **PCA** performs worse than other feature selection approaches.









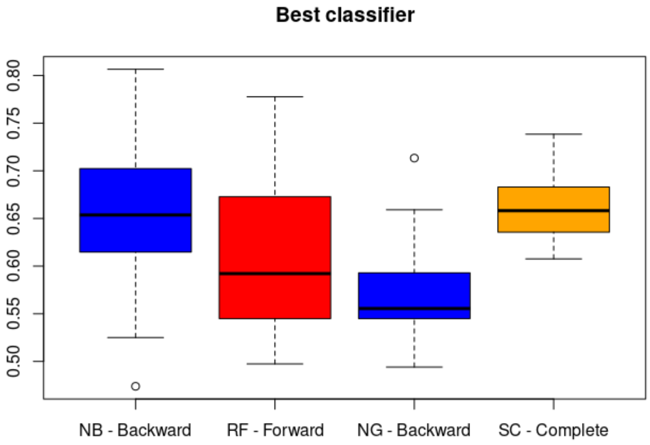
Comparison of classifiers

Every method of classification and clustering is compared with the different feature selections approaches, using:

- **Friedman test:** nonparametric test to compare if different methods have a significantly different performance. Tests the null hypothesis that different methods have the same performance.
- **Nemenyi test for multiple comparisons:** this nonparametric test is applied if the Friedman test rejects the null hypothesis.

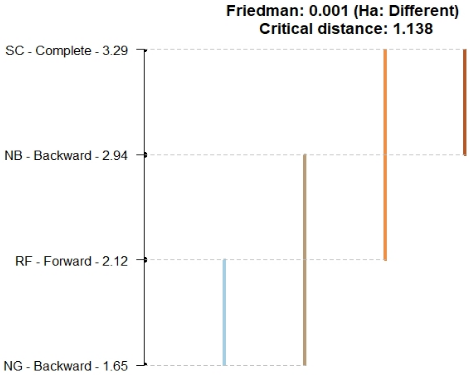
Comparison of classifiers: AUC

Naive Bayes and **Spectral Clustering** have the best AUC performance. **Random Forest** achieves slightly lower results, but has higher variability. **Neural Gas Clustering** has the lowest AUC values.



Comparison of classifiers: AUC

Spectral Clustering and **Naive Bayes** rankings do not differ by the Nemenyi critical distance. **Random Forest** ranking barely exceeds the critical distance and so performs as well as **Naive Bayes,** but not as well as **Spectral Clustering. Neural Gas Clustering** exceeds by far the critical distance: it achieves the worst performance.



Conclusion

RQ1. Do current unsupervised methods perform better or worse than the supervised ones?

- **Spectral Clustering** achieves average **0.66 AUC** on the complete datasets.
- **Naive Bayes** achieves average 0.65 AUC with backward selection.
- **Random Forest** achieves average 0.61 AUC with forward selection.
- However, their performances are **not significantly different**. The only method performing **worse** than the others is **Neural Gas Clustering.**

Conclusion

RQ2. Which feature selection method is the best for supervised methods?

- **Backward** selection is the best for **Naive Bayes.**
- **Forward** selection is the best for **Random Forest.**
- However, different **feature selection approaches do not influence significantly the performance of supervised methods.**

Conclusion

RQ3. Which feature selection method is the best for unsupervised methods?

- **Backward** selection is the best for **Neural Gas Clustering.**
- Using the **complete datasets** is the best approach for **Spectral Clustering.**
- All the feature selection approaches perform similarly, except for **PCA** which performs the worst.

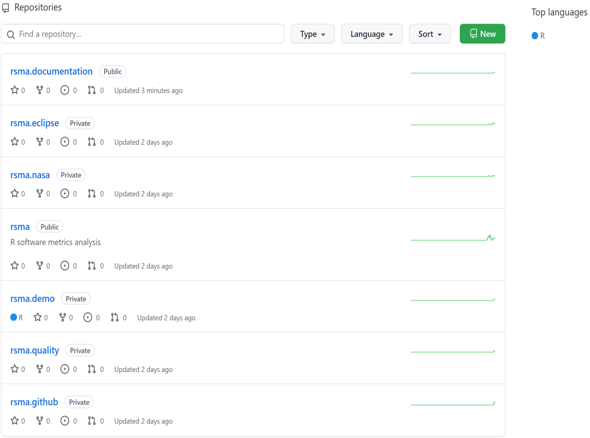
Thanks for your attention

For more information, see the publications:

- E. Ronchieri, M. Canaparo, G. Bertaccini - A framework to conduct feature selection on software metric datasets, December 2021, in Proceedings of SDPS 2021, Workshop on Smart Pervasive Computing, `https://sdpsnet.org/sdps/documents/sdps-2021/SDPS%202021%20Proceedings.pdf#nameddest=15`.
- E. Ronchieri, M. Canaparo, G. Bertaccini - Software defect prediction: A study on software metrics using statistical and machine learning methods, March 2022, presented at ISGC 2022, paper under submission.

R framework

R framework at `https://github.com/rsma-defect-prediction`.



Appendix

Number of features selected by different feature selection approaches.

Dataset	Complete	Backward	Forward	LASSO	PCA
Eclipse	18	10	7	5	11
Equinox	18	7	6	9	10
Lucene	18	9	5	15	11
Mylyn	18	9	7	11	11
PDE	18	**13**	7	**14**	10
CM1	38	**13**	10	9	10
MW2	38	13	7	8	11
PC1	38	**19**	14	**18**	11
PC3	38	**23**	10	**26**	11
PC4	38	**21**	19	**29**	14
JM1	22	12	14	**21**	9
KC1	22	**13**	12	**16**	8
KC3	40	**14**	7	**12**	9
MC2	40	**20**	5	8	10
MC1	39	**16**	9	**15**	15
PC5	39	**22**	15	**27**	14
PC2	37	**11**	6	**8**	8