Software Defect Analysis and Prediction

Ali Arefi (oarefi)
Jalaj Rastogi (jrastogi)

Motivation

Good code contains less errors, and as code becomes more complex and dynamic, it becomes more difficult to assess its performance and likelihood to fail. We need an easy and fast way to test and understand code as well as easier maintainability. McCabe and Halstead metrics can be used to calculate code complexity and allow a developer to design better quality software that won't fail or have low risks of failing. This is especially important for safety-critical applications that are required to have high standards by governments (industries like automotive, manufacturing, aerospace, medical devices, and power stations all need software metrics to help them make decisions about redesigns).

McCabe and Halstead metrics comprise of parameters which can be calculated from the code such as number of unique operators, lines of code, etc. Measurement of software complexity by Verisoft has given some formulas for calculating ideal values/ranges for these parameters which are essential for good software design.

Intent Summary:

1. We utilize the Verisoft parameters to see which programs fall within redesign classification, we explore the relationship between programs designated as redesign and programs that actually failed(defected)
2. Furthermore we will use a **RandomForrestClassifer** to train a software defect prediction system that will be able to classify new points like likely defective or not based on previous training data
3. Once optimized to an acceptable performance through validation, we will perform feature importance measures to see which features are more important in the models prediction performance.

Data Manipulation

1. Extract Data
2. Merge Data
3. Data Preprocessing

Extract Data

The datasets used in this project were in .arff format and were converted to csv format for ease of use. The column names in arff files are attached with “@attribute” and after all the columns the data is preceded by “@data”. We wrote a function that converted an arff file in a list of lines where the first line consisted of the column names and the remaining lines consisted of data rows. This list was then written into a csv file.

Merge Data

The datasets were merged using outer join. PC3 and PC4 had similar columns except the ‘c’ column in PC4 and ‘Defective’ column in PC3. Both these columns referred to the same attribute i.e whether there was a defect or not but had different column labels and different values representing the presence of defect. After merging the resulting dataframe had 2535 rows and 40 columns.The NaN values were replaced by empty strings and values in both the columns were concatenated and assigned to a new column ‘defects’. This was followed by replacing ‘True’ and ‘Y’ values with ‘1’, and ‘False’ and ‘N’ values with ‘0’ in the ‘defect’ column. Similarly MC1 dataset was merged into the resulting dataframe which now had 4523 rows and 41 columns.

Data Preprocessing

We wanted to add another feature named ‘Redesign Status’ which would tell whether a particular software module (a row in the dataset) needed to be re-implemented or not. In order to create this column we selected the features from the dataset for which we either had some ideal values or we could calculate by referencing the document : Measurement of software complexity provided by Verisoft. The following values and formulas were used to set condition to create the ‘Redesign Status’ column:

1. **CYCLOMATIC_COMPLEXITY** should be less than 15. The presence of constructs such as if.., for loop, while loop, case.., increases the cyclomatic number and hence increases the possible execution paths of the software.
2. **HALSTEAD_VOLUME(V)** should be between 20 and 1000. It describes the size of implementation of an algorithm and is dependent on the number of operators and operands.
3. **HALSTEAD_ERROR_EST** should be less than 2. It is the estimate for the number of errors in the software module.
4. **HALSTEAD_EFFORT(E)** should be less than 464758.0. This value was calculated using the formula: B = E^(⅔)/3000, where B = **HALSTEAD_ERROR_EST**. It refers to the effort to implement or understand the program.
1. **HALSTEAD_PROG_TIME(T)** should be less than 25820. This value was calculated using the formula: **T = E/18.** It refers to the time to implement or understand the program
2. **HALSTEAD_DIFFICULTY(D)** should be less than 23237.9. This value was calculated using the formula: **E = V*D.** It refers to the difficulty or error proneness of the program.
3. **LOC_TOTAL** should be less than 400. It refers to the total line count of the software module.

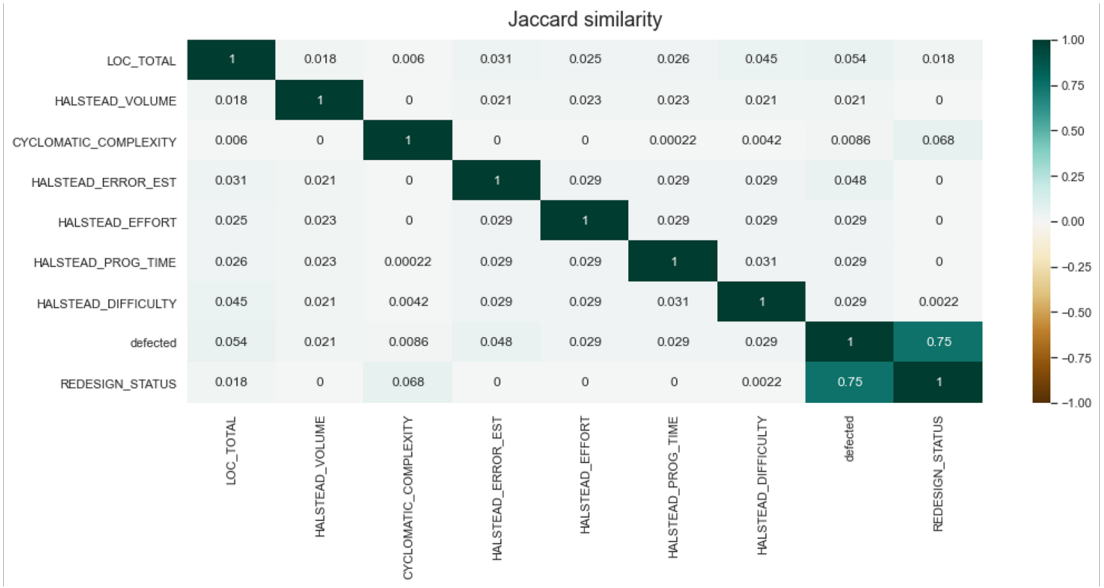
Based on the above conditions we created the ‘Redesign Status’ where if a software module satisfied the above conditions the value was set to 0 and otherwise 1.

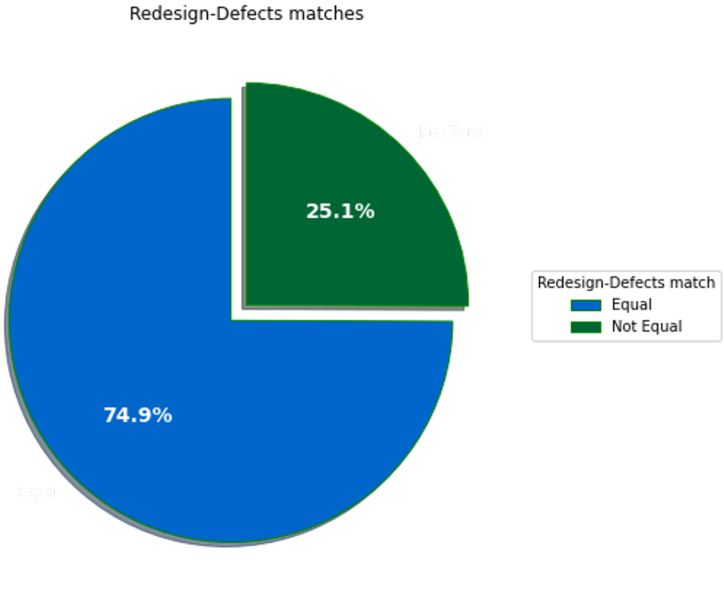
Data Visualization and Analysis

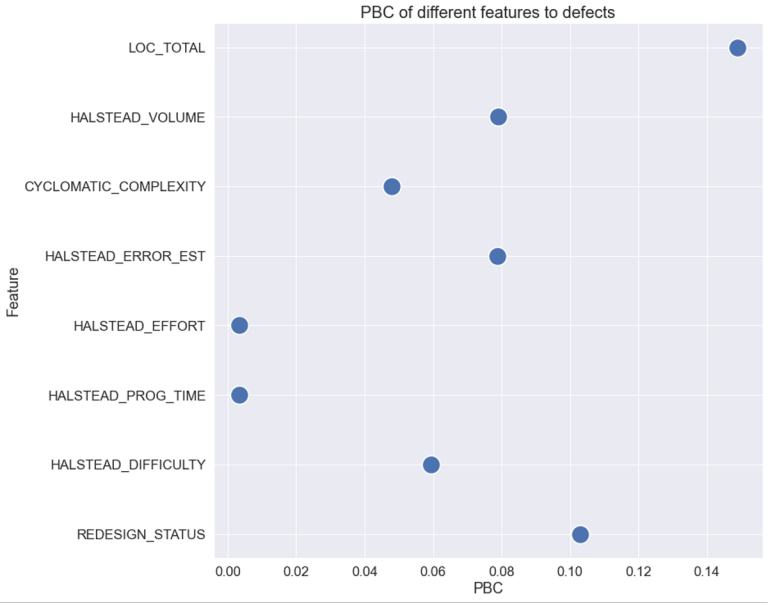
Defects and Resign

Here we measure how similar features are to one another using Jaccard similarity, the ratio of the number of matches between any two features over the number of total value of both those features. We see that Redesign and defects have a 75% similarity. So just applying the verisoft metric recommendations, we have a 75% chance of catching defective software with less than 4,500 programs under evaluation.

Defects and Resign







Point Biserial Correlation

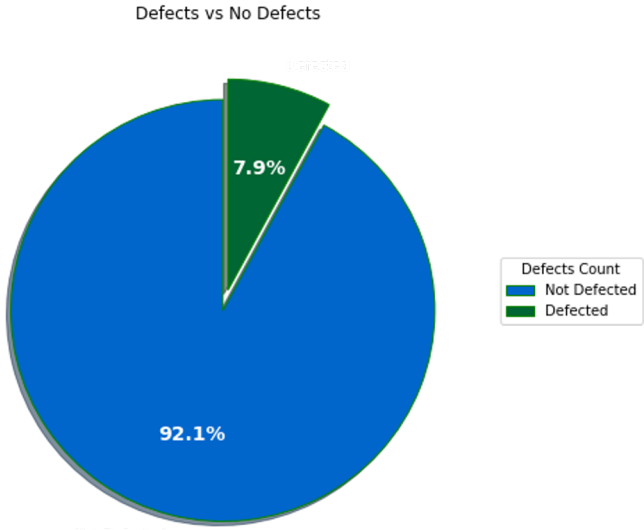
This method is used when we need to measure correlation between binary features and continuous features. Our defects are binary and our features are continuous values, therefore PBC is appropriate in this case.
Here we see that there is a slight correlation between our features and defects. The most being LOC or total lines of code in the program. Redesign status has high similarity with defects and is slightly more correlated than most of the other features. Halstead volume and Halstead errors or bugs, which are dependent on the total number of operators and operands and total number of distinct operators and operands, are more correlated with defects than the other features. Cyclomatic complexity is slightly correlated as well but not as much as the other features. PBC does not indicate positive or negative, just strength of correlation

Software defect prediction system

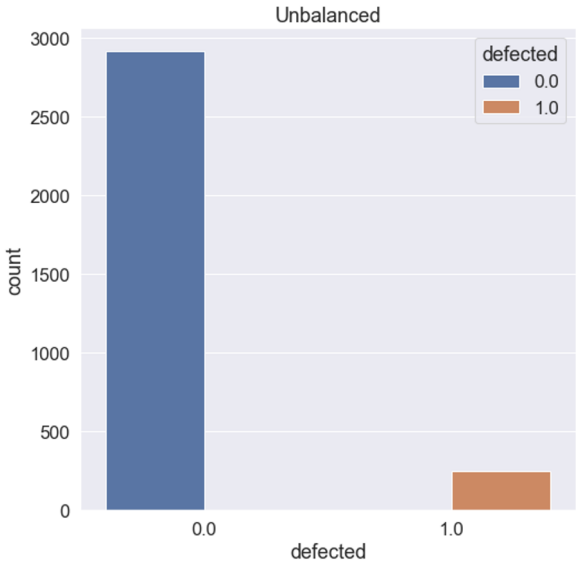
We build our system using an optimized RandomForrestClassifier, which uses decision tree classifiers on various sub-samples of the data, averaging them out to improve the predictive accuracy. A consequence of this is control over over-fitting. We optimize our the parameters of the model using RandomSearchCV

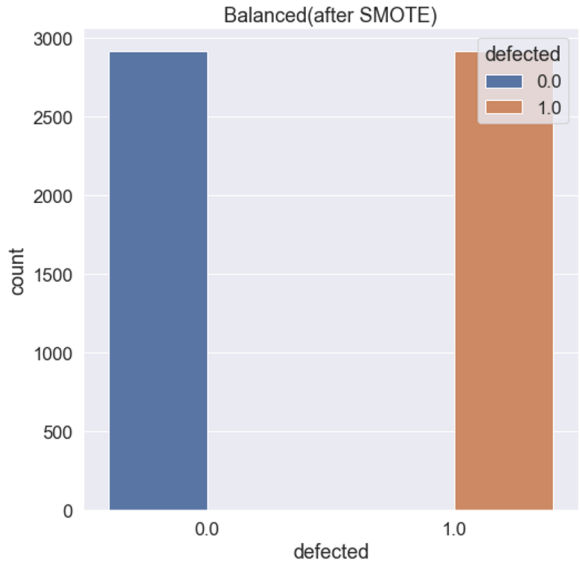
Issues: Unbalanced data set

An issue in most predictive models is an unbalanced dataset, in our case, meaning defects are much less than non-defects. This becomes an issue when the system is trying to learn how to predict.



(**unbalanced continued**) The classification algorithm will tend to categorize into the class with more instances, and as it learns the majority class better. The algorithm will not learn how the minority class is different, failing to understand the underlying patterns that allow it to distinguish between them.





Hyperparameter optimization

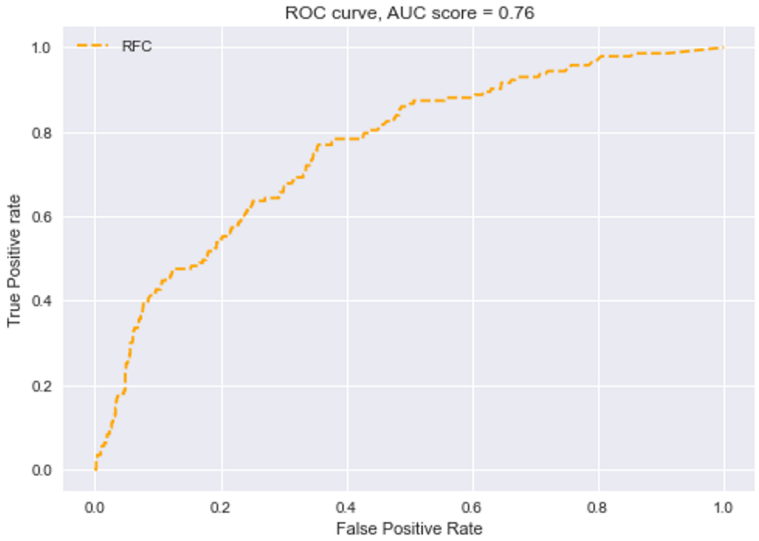
We optimize our parameters using RandomSearchCV, which outputs the best parameters to use to fit our feature training samples and our target training samples.

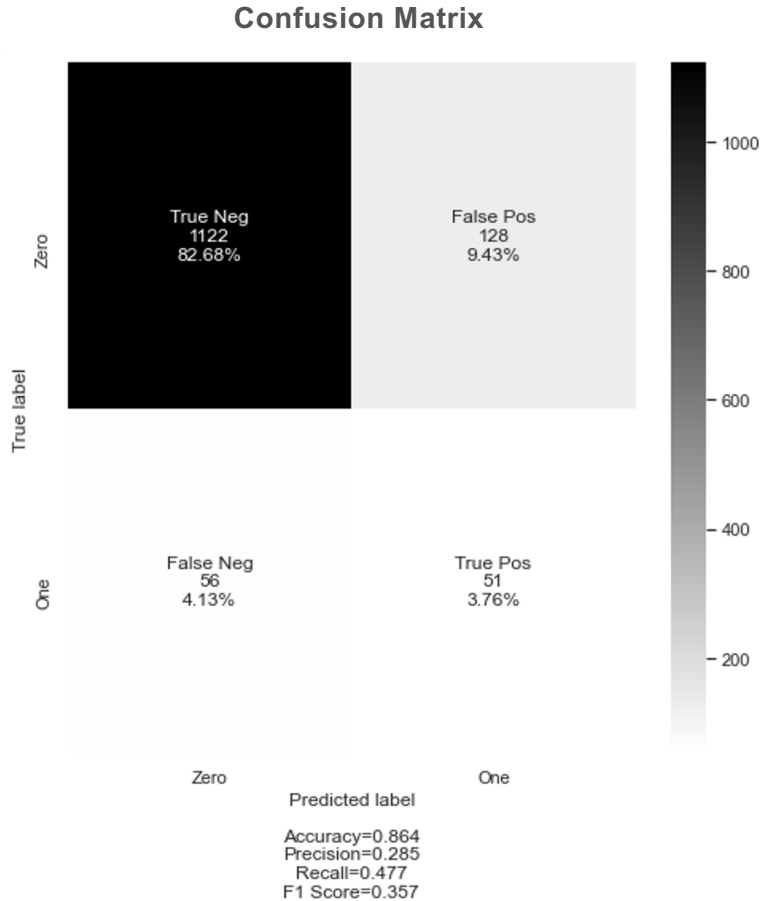
n_estimators	1400
min_samples_split	2
min_samples_leaf	1
max_features	sqrt
max_depth	110
bootstrap	False

Performance of model

The model is able to predict the majority class relatively well, but has low precision(false positives are high) even after oversampling.

ROC-AUC

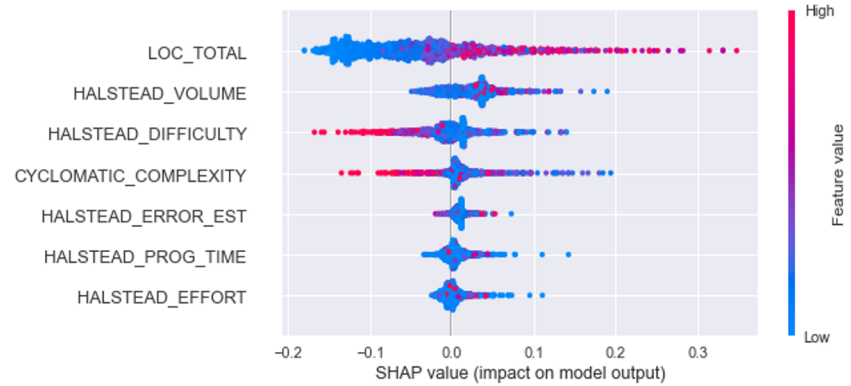




The Receiver Operator Characteristic (ROC) is used for binary classification problems by plotting a probability curve of the true positive rate against false positive rate at various threshold values. The Area Under the Curve (AUC) measures the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. AUC scores range from 1 to 0, but the focus is from the range of 0.5 to 1. A score of 0.5 indicates that the classifier is randomly guessing . Values from 0.9 to 1.0 are excellent. Our value of 0.76 is an adequate classifier but not exceptional.

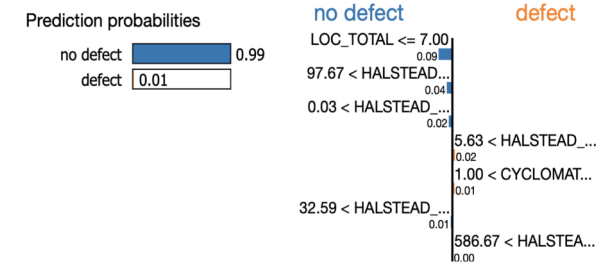
Feature importance

To get an idea of which features are most important for our prediction, we do certain tests for feature importance. This shap value plot shows the training data, represented as dots, where each one is associated with a high(defect) prediction or low(not defected) prediction. We see that LOC_TOTAL is correlated with the defect classification while Cyclomatic complexity and Halstead difficulty is is negatively correlated with defect classification. This means that the Difficulty, which measures how many times an operand has been used in a program, along with Cyc. Complexity, which measure the possible execution paths of the program, have more of a negative correlation on defect classification.



Conclusion

Here we investigated the relationships between defected programs, programs marked for redesign by recommendation(higher likelihood of failure), and features that are fundamental software metrics, metrics used for redesign recommendations. We saw by IQR, that redesigns and defects are all outliers, matching in value for 75% of all instances. We made a software defect prediction system of adequate performance. Out of all the important features of this system, a recurring theme was the positive correlation of total lines of code and volume of operands and operators to defects. Cyclomatic complexity has some instances of a negative correlation with defects. Although cyclomatic complexity is associated with more defects, there is no proof for this in the field and our model might indicate this as well.



Feature	Value
LOC_TOTAL	7.00
HALSTEAD_VOLUME	108.42
HALSTEAD_ERROR_EST	0.04
HALSTEAD_DIFFICULTY	7.86
CYCLOMATIC_COMPLEXITY	3.00
HALSTEAD_PROG_TIME	47.33
HALSTEAD_EFFORT	851.86

Statement of Work

ALL : Report Writing

Ali Arefi : Prediction system and feature importance

Jalaj Rastogi : Data Manipulation, report design and editing

Resources:

1. **Verifysoft.com. 2012. [online] Available at: <https://verifysoft.com/en_software_complexity_metrics.pdf> [Accessed 25 January 2022].**
2. **https://en.wikipedia.org/wiki/Cyclomatic_complexity**
3. **https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74**
4. **https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea**

Data Sources :

Name	Source	Format	Size	Important Variables
MC1	mc1.arff	.arff	1988 rows and 40 columns	Halstead metrics, Cyclomatic Complexity
PC3	pc3.arff	.arff	1077 rows and 39 columns	Halstead metrics, Cyclomatic Complexity
PC4	pc4.arff	.arff	1458 rows and 39 columns	Halstead metrics, Cyclomatic Complexity