CYBR 520 – Homework #2 (Individual) –Feature Selection and Supervised Machine Learning in Software Bug Prediction.

**Logan Johnson**

# Instructions and submission:

# Please read the following document and submit your answers in this file after each question below, where applicable. Submit your code as a separate Jupyter Notebook, named HW2, within the Homework folder in your GitHub repository. Do not include any code in this answer file.

# Problem statement:

You are a software security analyst who has been notified of a vulnerability in Apache Synapse [1] caused by a software bug. You have been requested to assess various supervised machine learning algorithms for their ability to help detect software bugs. You have been provided with a dataset containing the names of Java source code files and their extracted features (i.e., static code metrics like WMC - Max\_CC and change metrics like SUM\_LOC+ - REF). These features describe the characteristics and modifications made to the source code files during bug fixes [2]. For instance, the file “ProxyServiceMessageReceiver.java” has an RFC of 62 and is labeled as buggy (TRUE), indicating it is a buggy file.

Using the code give on GitHub and this dataset, you need to find the best classification model with the best set of features (i.e., static, change, a combination of both, manually selected features or using any other selection features techniques).

The current code runs a classification model with the following settings:

1. Using all features (i.e., static and change).
2. Uses Logistic Regression as the machine learning algorithm. (You will need to calculate False Positive Rate -FPR- yourself).
3. Set 20% of the data for testing.

Based on the confusion matrix, the results are given in Table 1.

Table 1: Results of classification using LogReg

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Features** | **Testing%** | **Accuracy** | **Precision** | **Recall** | **FPR** | **MCC** |
| Logistic Regression | All Features | 20% | 87.59% | 74.29% | 74.29% | 8.18% | 66.10% |

You are to run several experiments using different sets of features and compare the results of these models and recommend the best model for software bug prediction.

# Objectives:

1. Use the code on GitHub to run different experiments to detect buggy software files.
2. Run selected algorithm using the full sets of features, static code metrics only, change metrics only, a combination of both, and using a feature selection technique or manually selected features.
3. Evaluate different models using Accuracy, Precision, Recall, FPR, and MCC.
4. Recommend best model.

# Deliverables:

1. This document with your answers (when applicable) below each question.
2. Jupyter Notebook name HW2 under your HomeWork folder on GitHub with the code used to answer these questions. Provide a comment of the question number before each cell please.
3. An eCampus note stating your have submitted your work.

Instructions:

1. Download the Jupyter notebook HW2 and the Synapse.csv dataset from the CYBR520 GitHub repository.
2. Save the notebook and the dataset on your local repository under the homeworks folder.
   1. After activating the correct Anaconda environment, use this command below to open the Jupyter notebook from the Anaconda terminal:  
       jupyter notebook –notebook-dir=” the local location of your homework folder on your Git repo”
3. Run the code and ensure you have all the libraries needed. Do not make any changes in the code when stated. You are free to change the rest as you need.
4. The current setting runs the experiment given in the highlighted row of Table 2 in the Questions section. (note that your results might be slightly different, it is ok).
5. Modify the code as needed and run the classification models to fill out all elements in of Table 2.
   1. For example, for the second row, you will have to ensure you only select the static code features in the code after you import the data from the csv file and run the code again.
6. Run the code using Logistic regression with the each of following settings:
   1. Using all features (current setting of the code)
   2. Using Static features only
   3. Using change features only
   4. Use a feature selection technique to obtain the most important 10 features or select the top 10 features based on any analysis you desire. Elaborate on your choice.
7. Re-run the models using another algorithm of your choice (i.e., commented out in the provided code).

Questions:

1. Report all results of all models in the empty cells in Table 2. **[25 points]**

Table 2: Results of classification models using different sets of features.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Features | Testing% | Accuracy | Precision | Recall | FPR | MCC |
| LogisticRegression | All Features | 20% | 87.59% | 74.29% | 74.29% | 8.18% | 66.10% |
| LogisticRegression | Static Only | 20% | 75.86% | 50% | 11.43% | 3.64% | 14.60% |
| LogisticRegression | Change Only | 20% | 85.52% | 85% | 48.57% | 2.73% | 56.89 |
| LogisticRegression | your selected features | 20% | 83.45% | 82.35% | 40% | 2.73% | 49.58% |
| Algorithm of your choice (state which) SVC | All Features | 20% | 89.66% | 73.81% | 88.57% | 10% | 74.12% |
| Algorithm of your choice (state which) SVC | Static Only | 20% | 73.79% | 36.36% | 11.43% | 6.36% | 81.19% |
| Algorithm of your choice (state which) SVC | Change Only | 20% | 84.83% | 80.95% | 48.57% | 3.64% | 54.64% |
| Algorithm of your choice (state which) SVC | your selected features | 20% | 81.83% | 78.57% | 31.43% | 2.73% | 41.58% |

1. Conduct some basic EDA on the dataset. Provide only two charts of your choice and explain what you are viewing.

A diagram of a graph

Description automatically generated with medium confidence A diagram of a graph

Description automatically generated with medium confidence

**On the left was the confusion matrix for change features using KNeighbors which yielded 7.27% FPR. On the right was an FPR of 10% using SVC and all features. I believe adding more features adds to the number of false positives you receive, but increases accuracy.**

* 1. Show the distribution of buggy files vs non buggy files. **[5 points]**

1. Which set of features (i.e., all, static only, change only, selected features of your own) achieved the best classification accuracy when used with logistic regression?

**For me, I achieved the highest accuracy when all features were selected.** **[10 points]**

* 1. Which features did you select? **[5 points]**

**I selected MAX\_CHRN, SUM\_CHRN, SUM\_LOC+, MAX\_LOC+, AMC, CC, MaX\_CC, SUM\_LOC-, LOC, & MAX\_LOC-**

* 1. Why did you select those? Justify your answer and elaborate on why you selected those features. **[10 points]**

**I selected them because on the correlation matrix with Bugs included, they had the most values closest to 1 which were characteristics that had almost perfect positive correlation.**

1. From the code, add another algorithm of your choice and repeat step 3 and provide an answer to the same question in Step 3.  **[10 points]**

**I again achieved the highest accuracy when all features were selected. This time I used KNeighbors.**

1. Which of all these models was the best? **[5 points]**

**In my research it was SVC using all features that was the best performing.**

* 1. What are downsides of choosing this model compared to other models?

**In talking specifically about the downsides of SVC over Logistic Regression, in my research I discovered that SVC is best utilized when a dataset is very small. The computational efforts that go into SVC are way more complex and Logistic Regression is normally a more balanced choice. However, in this case, our data set was not too large which is what helped SVC achieve the best results during my runs with the data.**

* 1. Which performance metric did you use to justify your choice? Why did you not choose others?  **[10 points]**

**I chose to focus mainly on accuracy, precision, and false positive rate because I felt that these were the most important when it comes to truly discovering which files meet the requirements for discovering software bugs. It is crucial to not tamper with files that are perfectly fine and a low false positive and high accuracy rate means that you are less likely to label a good file as a bug file. However, it seems as though the higher the accuracy is, the higher your FPR goes as well.**

1. How does the choice of the features affect the performance of the classification models? Provide evidence. **[10 points]**

**As you can see in the tables above, the most features we used was all, and with 3 different methods in Logistic Regression, SVC, and KNeighbor, the numbers for accuracy and precision were highest. I believe that adding more features simply adds to the accuracy, as all aspects of the data are being analyzed, not just static, change, or custom features.**

1. Submit your code on GitHub **[10 points]**

**Bonus**:

Provide 5 line charts that show each of the performance metrics (i.e., Accuracy, Precision, Recall…etc.) for each of the experiments. **[10 points]**

# Bibliography

[1] (n.d.). Retrieved from Apache Synapse: https://projects.apache.org/project.html?synapse

[2] Ahmad, Mohammad Jamil, Katerina Goseva-Popstojanova, and Robyn R. Lutz. "The untold impact of learning approaches on software fault-proneness predictions: an analysis of temporal aspects." *Empirical Software Engineering* 29.4 (2024): 87. **(This is also avaialble on the GitHub repor of this class).**