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

# 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% | 78.62% | 62.50% | 28.57% | 5.45% | 31.57% |
| LogisticRegression | Change Only | 20% | 85.52% | 71.88% | 65.71% | 8.18% | 59.36% |
| LogisticRegression | your selected features | 20% | 91.03% | 80.56% | 82.86% | 6.36% | 75.77% |
| RandomForest | All Features | 20% | 93.79% | 84.21% | 91.43% | 5.45% | 83.66% |
| RandomForest | Static Only | 20% | 73.10% | 42.31% | 31.43% | 13.64% | 19.85% |
| RandomForest | Change Only | 20% | 94.48% | 84.62% | 94.29% | 5.45% | 85.73% |
| RandomForest | your selected features | 20% | 93.79% | 80.95% | 97.14% | 7.27% | 84.78% |

1. Conduct some basic EDA on the dataset. Provide only two charts of your choice and explain what you are viewing.
   1. Show the distribution of buggy files vs non buggy files. **[5 points]**
2. 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? **[10 points]: The Selected Features set achieved the best classification accuracy with Logistic Regression, scoring 91.03% accuracy.**
   1. Which features did you select?**:**

The top 10 selected features identified using Recursive Feature Elimination (RFE) are:

1. RFC (Response for Class)
2. CBO (Coupling Between Objects)
3. WMC (Weighted Methods per Class)
4. SUM\_LOC+ (Sum of Lines of Code added)
5. MAX\_LOC+ (Maximum Lines of Code added)
6. REF (Number of References)
7. NO\_REV (Number of Revisions)
8. SUM\_CHRN (Sum of Changes in the Current Revision Number)
9. MAX\_SET (Maximum Set Size)
10. Age (Age of the file in weeks)
    1. Why did you select those? Justify your answer and elaborate on why you selected those features. **:**
       1. RFE Output
          1. The RFE process pinpointed these features as the most predictive for the Logistic Regression model. Basically, they had the biggest impact on the model's performance.
       2. Interpretability
          1. The selected features combine static metrics (like RFC, CBO, and WMC) and change-based metrics (such as SUM\_LOC+, REF, and NO\_REV). This mix gives a well-rounded view of the dataset.
       3. Improved Accuracy
          1. By using these features, the model hit its highest accuracy: 91.03%. That’s better than any other feature set tested.
11. From the code, add another algorithm of your choice and repeat step 3 and provide answer to the same question in Step 3.  **[10 points]**
12. Which of all these models was the best? **:**

The Random Forest model, using only change features, came out on top with an impressive classification accuracy of 94.48%.

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

**Computational Complexity**

Random Forests need more computational power than Logistic Regression because they train a bunch of decision trees and then combine their results. This process can take longer, especially if you're working with a big dataset, it might slow down both training and making predictions.

**Interpretability**

Here’s the trade-off, Random Forests are powerful but work like a black box it’s tough to understand how they connect features to predictions. On the other hand, Logistic Regression is straightforward. It gives you clear feature importance through its coefficients, making it much easier to interpret.

**Overfitting Risk**

Random Forests try to avoid overfitting by using methods like bootstrapping (sampling with replacement). Still, they’re not immune. If your dataset isn’t well-prepped, like being imbalanced, overfitting can creep in.

**Feature Importance Dependency**

Random Forests shine when you feed them the right features. But here’s the catch: they don’t handle irrelevant features well. If you include too much noise, even their built-in robustness can’t always save accuracy.

* 1. Which performance metric did you use to justify your choice? Why did you not choose others?**:**

Chosen Metrics: Accuracy and Matthews Correlation Coefficient (MCC)

* **Accuracy** gives you a big-picture look at how well the model predicts both buggy and non-buggy files overall. It’s great for understanding the model’s general correctness.
* **MCC** is a more balanced metric. It takes into account all parts of the confusion matrix—true positives, true negatives, false positives, and false negatives. This makes it especially useful when your dataset is imbalanced.

Why Not Just Precision or Recall?

* **Precision** zooms in on the positive predictions but ignores how well the model handles true negatives. That means it’s not telling the full story.
* **Recall** is all about catching every positive, but that focus can come at a cost—it might boost false positives.

1. How does the choice of the features affect the performance of the classification models? Provide evidence:

The Impact of Feature Selection:

* **Using All Features**: Including every feature might seem like a safe bet, but it often introduces noise. This noise hurts the model’s ability to generalize. For example, Logistic Regression’s accuracy with all features was **87.59%**—not bad, but it struggled to cut through the clutter.
* **Static** Fea**tures Only**: When the model only uses static features, it loses the ability to track dynamic changes, which limits its effectiveness. Accuracy drops to **78.62%**, showing that static data alone isn’t enough.
* **Change Features Only**: Here’s where things get interesting—focusing just on change-related metrics led to the **highest accuracy** for Random Forest at **94.48%**. Why? Because these features are great at spotting buggy files.
* **Selected Features**: Combining the best of static and change features strikes a balance. By removing irrelevant data, Logistic Regression’s accuracy jumped to **91.03%**, a noticeable improvement over using all features.

Evidence Highlights:

* Logistic Regression accuracy climbed from **87.59%** (All Features) to **91.03%** (Selected Features) after irrelevant features were removed.
* Random Forest hit **94.48%** when it relied on change features alone, showing that relevant, dynamic metrics are incredibly powerful indicators.

The Bottom Line:

* Choosing the right features is critical. Including too many irrelevant features adds noise, which weakens the model’s predictive power. On the other hand, selecting only the most relevant features—especially a mix of key static and dynamic ones—boosts accuracy and helps the model perform at its best. It’s all about finding that balance.

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).**