CSC522: Intermediate project report

Title: Are Software Metrics Good Predictors for Vulnerability Assessment of Android Applications?
Team: Repo Miners

1 Introduction

Introduction With the increase in popularity and use of Android apps, security and privacy issues of Android apps are becoming a concern [1][3]. Android app developers can benefit in writing better quality apps if a study can systematically analyze which software engineering issues such as coding conventions and software metric contribute to Android app vulnerability. Prior studies have shown how software related metrics can be used to predict software defects. We draw inspiration from these studies and identify the possibility of applying data mining techniques to predict vulnerability in Android apps. In this project we aim to predict vulnerability risk score for each version of apps from different features from the dataset such as number of commits, number of contributors, number of classes, number of lines per code, and number of functions.

2 Dataset

The dataset [2] is a 87. 5 MB SQLite database containing software related metrics data and vulnerability data of 1179 open source Android apps that has 4416 different versions. The dataset contains a vulnerability risk score for each of the versions of the 1179 applications. The dataset also contains software metrics for each version of the 1179 apps such as number of files, number of directories, file complexity, and number of violations.

3 Methodology

In this work, we focus our efforts on predicting how much Android applications are susceptible to vulnerability. The dataset provides a 'risk score' for all the Android application versions. We hypothesize that raw risk scores might not be helpful Android applications. Instead of predicting the risk scores we categorize the scores in two categories: high and low (defined as per the median of scores in our labeled data-set). For categorizing the scores we use hierarchical clustering.Hierarchical clustering categorizes the all the scores into two categories:high and low. Next, we use the following steps to classify if an Android application is highly or less susceptible to vulnerability, by applying the following steps:

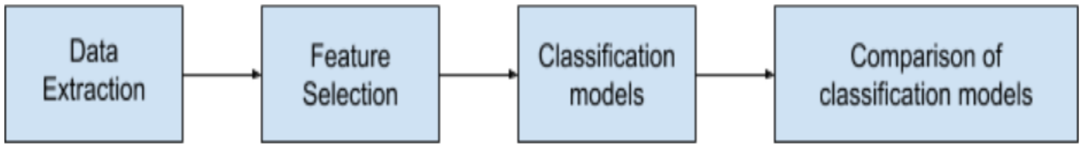

Figure 1: Major steps of the study

For this purpose, we applied classification models as mentioned below on our labeled dataset:

- Feature Selection/Reduction: We observed 20 software-metric related features in the dataset that can be used for classification. We have used simple logistic regression model to determine the features that will be used to classify the high and low risk applications. After applying logistic regression if the observed correlation coefficients were non-zero for a certain feature, then we consider that feature for classification. After applying simple logistic regression model we observed that all the features had non-zero correlation coefficients, and hence we included all the 20 features for classification.
- Classification models: Three classification models were applied on the dataset : SVM, Random Forests and Decision Trees using the sklearn python library.
- Validation: For the purpose of validation, we compared the Precision, Recall and Area under the ROC curve measures for different sizes of the training data sets (10

4 Findings

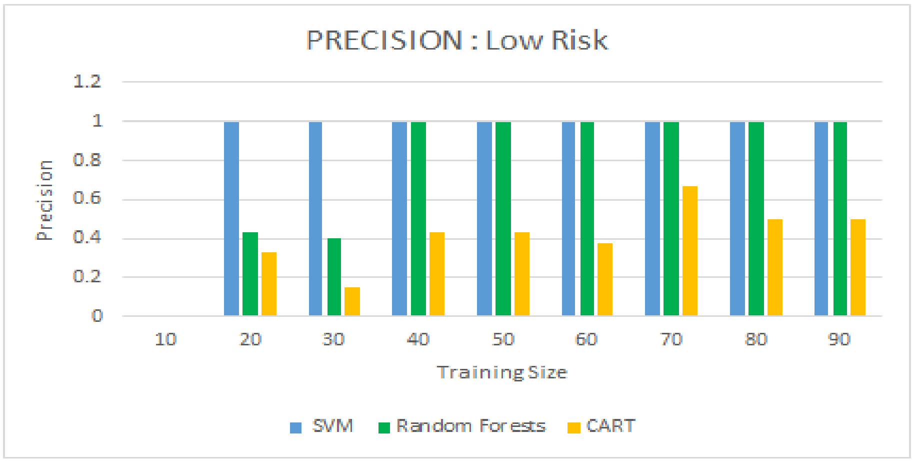

Figure 2: Precision for low risk android versions

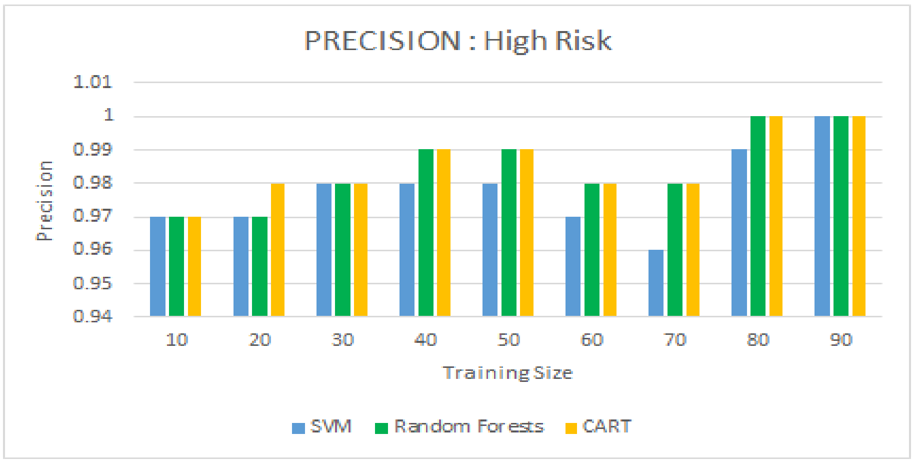

Figure 3: Precision for high risk android versions

Figure 2 and 3 compare the precision for low and high risk clusters respectively. From Figure 2, we can conclude that for the low-risk cluster, SVM and Random Forests classification models have higher precision in comparison to CART classification model.

Whereas, Figure 3 represents that for the high-risk cluster, CART and Random Forest provide a higher precision than SVM classification model. Thus, we can conclude that overall Random Forest classification model performs the best in terms of precision.

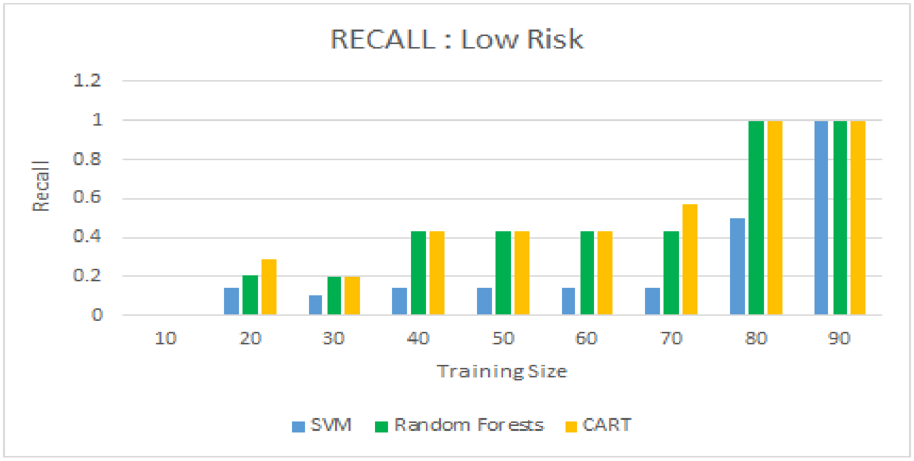

Figure 4: Recall for low risk android versions

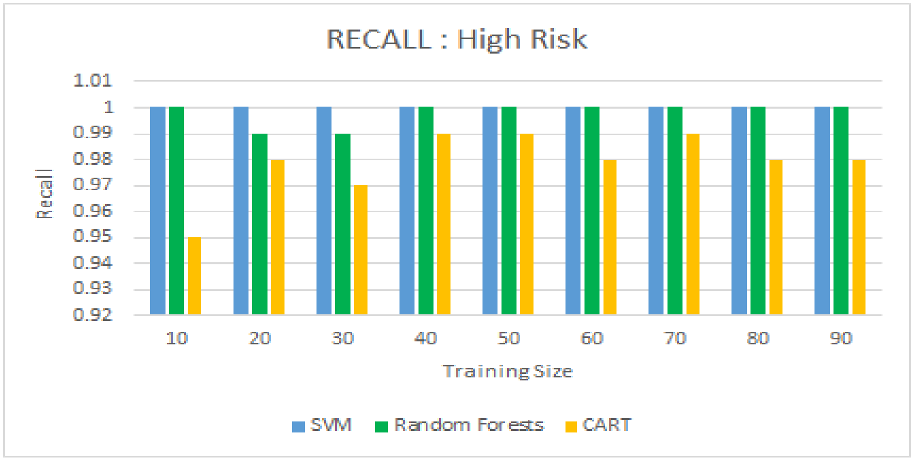

Figure 5: Recall for high risk android versions

Figure 4 and 5 compare the recall for low and high risk clusters respectively. Similarly, from Figure 4, it can be observed that for the low-risk cluster, CART and Random Forests classification models have higher recall value in comparison to SVM classification model.

Whereas, Figure 5 shows that for the high-risk cluster, SVM and Random Forest provide a higher recall value than CART classification model. Thus, through these observations it can be concluded that Random Forest classification model performs the best in terms of recall too.

Figure 6 displays the area under ROC curve for all three classification models. On observing this graph ,it can be seen that Random Forest and CART reach the best possible value (1.0) with a smaller training size compared to that of SVM.

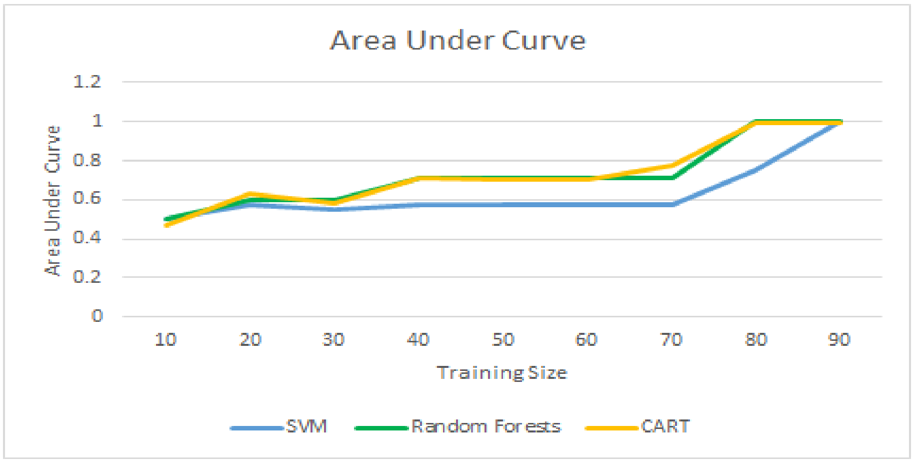

Figure 6: AUC for the three classification models

References

[1] Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. 2012. CHEX: statically vetting Android apps for component hijacking vulnerabilities. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). ACM, New York, NY, USA, 229-240.

[2] Daniel E. Krutz, Mehdi Mirakhorli, Samuel A. Malachowsky, Andres Ruiz, Jacob Peterson, Andrew Filipski, and Jared Smith. 2015. A dataset of open-source Android applications. In Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15). IEEE Press, Piscataway, NJ, USA, 522-525.

[3] http://www.scmagazine.com/researcher-discovers-thousands-of-vulnerable-apps/article/410418/