Selina Narain, Neelam Boywah, Zoya Haq
DTSC 870 - Masters Project - Fall 2023
Advisor: Professor Dr. Wenjia Li

Progress Report 4

Timeline: November 1st, 2023 - November 15th, 2023

Accomplishments: What did you accomplish?

Research Topic Idea:

- Comparing machine learning and deep learning algorithms for accuracy and efficiency in detecting malware in android applications.
- Applying adversarial attacks on machine learning models.

Research:

X-ANOVA Ranked Features for Android Malware Analysis

- Approach: Statistical analysis to classify android malware.
- Optimal features taken from large attribute space
- The 3 features that are extracted from each Android.apk file are opcodes, methods and permissions.
- ANOVA (Analysis of Variance) has been modified to X-ANOVA where features are ranked based on their variance in malware and benign training sets.
- X-ANOVA is also used for dimensionality reduction in order to minimize errors.
- In data pre-processing, 156 opcodes, 69 methods, and 78 permissions are all features that have high frequency in the maximum number of instances.
- In Feature ranking, feature selection takes place where relevant features are determined to benefit the classification models in multiple ways. For example, better visuals of the data, utilizes less storage space, the retrieval data is more effective and because of the reduction in dimensions, the noise is eliminated and misclassification is avoided.
- Accuracy portrays the degree of correctness of the model.
- True Positive Rate (TPR) portrays the rate of malware correctly classified as malware.
- Precision portrays the relevant instance retrieved from the data sample.
- Classifiers built and tested to determine accuracy were J48, ADABoost and Random Forest.
- ADABoost resulted in 87.06% accuracy utilizing 64 features, J48 resulted in 85.82% accuracy utilizing 25 features and Random Forest resulted in 85.82% accuracy as well utilizing 50 features. ADABoost had the highest accuracy.

- When utilizing Bigram opcode, 800 bigram opcodes are started out with 14, 461 unique bigram opcodes. After utilizing X-ANOVA, 130 bigram opcodes are acquired.
- After testing the classifiers, ADABoost still has the highest accuracy of 88.30% with 55 opcodes. J48 resulted in 85.56% with 45 bigram opcodes and Random Forest resulted in 87.06% with 35 bigram opcodes.

Implementation for CICMaldroid 2020 Dataset:

- The necessary libraries were imported: numpy, pandas, sci-kit learn, seaborn, and matplotlib.
- We used the New Brunswick: CICMaldroid2020 Dataset for the initial implementation. 470 features extracted for 11,598 APK files (system calls, binders, and composite behaviors, 139 features extracted for 11,598 (system calls) and 50,621 features extracted (static information - intent actions, permissions, files, method tags, sensitive APIs, services, packages, receivers, etc).
- Sample sizes: Adware: 1,253, Banking: 2,100, SMS malware: 3,904, Riskware: 2,546, Benign: 1,795 = Total: 11,598
- Utilized Feature Selection by defining features and for the necessary features that were not being used, they were dropped.
- The data was then split into training and testing.
- ANOVA base feature selection was used.
- Some of the subsets of data include Adware, Banking, SMS malware, Riskware, and Benign.
- Features were then put into a data frame and shaped.
- The data set was scaled using Sci-kit Learn Standardscaler after dropping the feature ‘Class”. Then we trained and tested through the Naive Bayes Classifier, Random Forest Classifier, Logistic Regression and KNN.
- <u>Naive Bayes Statistics:</u>
    - Accuracy: 0.5892
    - Precision: 0.6936
    - Recall: 0.5892
    - F1-Score: 0.5462

- Random Forest parameters: n_estimators = 300 and random_state = 42.
- <u>Random Forest Statistics:</u>
    - Accuracy: 0.9422
    - Precision: 0.9432
    - Recall: 0.9422
    - F1-Score: 0.9421

- Logistic Regression parameters: max_iter = 1000000 and random_state = 42.
- Logistic Regression Statistics:
    - Accuracy: 0.8078
    - Precision: 0.8079
    - Recall: 0.8078
    - F1-Score: 0.8033

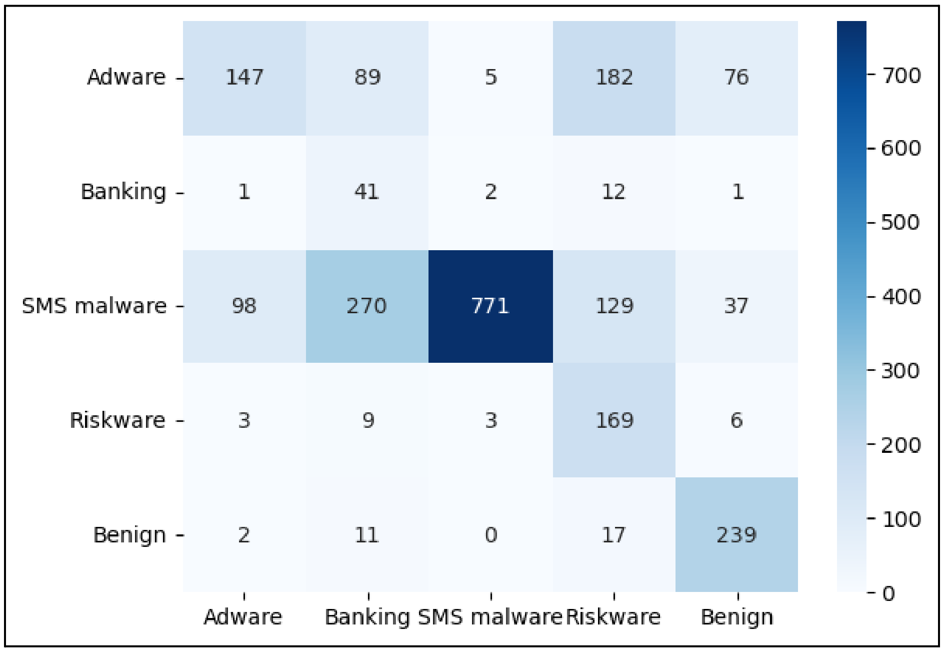
- The KNN model was used for cross-validation purposes and performed a 5 fold cross validation. The ranges of k values used to evaluate the model are 2, 3, 4, 5, 7, 9, 10 and 11. The parameter used in the KNN model is the n_neighbors = best_k which in this case was 3.
    - K-Nearest Neighbor Statistics:
        - Accuracy: 0.9022
        - Precision: 0.9059
        - Recall: 0.9022
        - F1-Score: 0.9025

Analysis:

- Naive Bayes Classifier is a supervised learning algorithm that assumes the conditional independence between each pair of features and the class variable value which is based on Bayes' theorem.
- Random Forest Classifier (supervised learning algorithm) is a set of decision tree classifiers where each one is trained from a data sample and is averaged to then determine the accuracy. The parameters utilized for our malware detection implementation are n_estimators (the number of trees) and random_state (controls randomness of the data sample).
- Logistic Regression Classifier is a supervised learning algorithm that helps to predict and differentiate between classes based on specific feature variables. The parameters utilized for our malware detection implementation are max_iter (maximum number of iterations) and random_state (controls randomness of the data sample).
- K-Nearest Neighbors is a supervised machine learning model that calculates the Euclidean distance to categorize data points to the corresponding class. The parameter utilized for our detection implementation is n_neighbors (number of neighbors).
- Based on these statistics, the Random Forest Classifier seems to be the best performing model for the task at hand so far.
- Accuracy is the overall correctness of a model. It isn't the only metric that is taken into consideration especially in imbalance datasets.

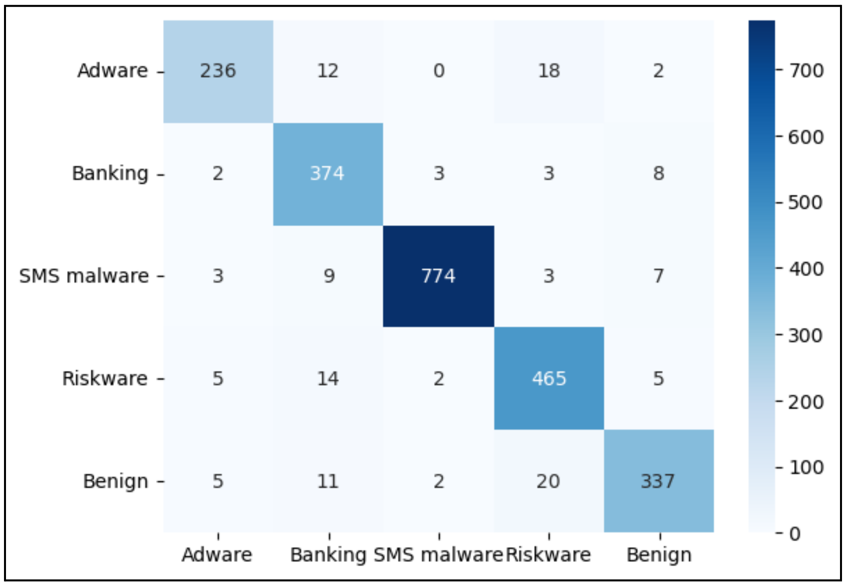
- Precision measures the accuracy of *positive* predictions. In this case, it is the percentage of malware that is actually malware.
- Recall is the sensitivity, it measures the ability of a model to capture *all* positive instances. For this project, the recall explicated the percentage of actual malware instances that were identified correctly by the model at hand.
- F-1 Score is the mean of precision and recall. It gives a balance between precision and recall. If there is a higher F-1 score, there is a better trade-off between precision and recall.
- Heat map displays the value for the amount of features (Adware, Banking, SMS Malware, Riskware, and Benign) that were classified correctly.
- Based on the aforementioned analysis, the Random Forest Classifier outperforms the other models in terms of the accuracy, precision, recall, and F-1 score and is the most suitable as of right now for the Android malware detection.

- <u>Naive Bayes Classification Heatmap</u>
    - Adware Features: 147
    - Banking Features: 41
    - SMS Malware Features: 771
    - Riskware Features: 169
    - Benign Features: 239

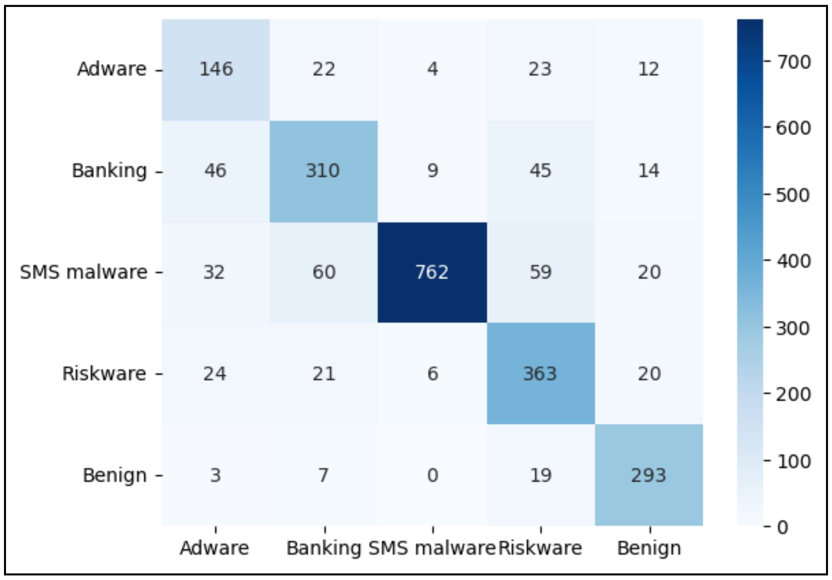


	Adware	Banking	SMS malware	Riskware	Benign
Adware	147	89	5	182	76
Banking	1	41	2	12	1
SMS malware	98	270	771	129	37
Riskware	3	9	3	169	6
Benign	2	11	0	17	239

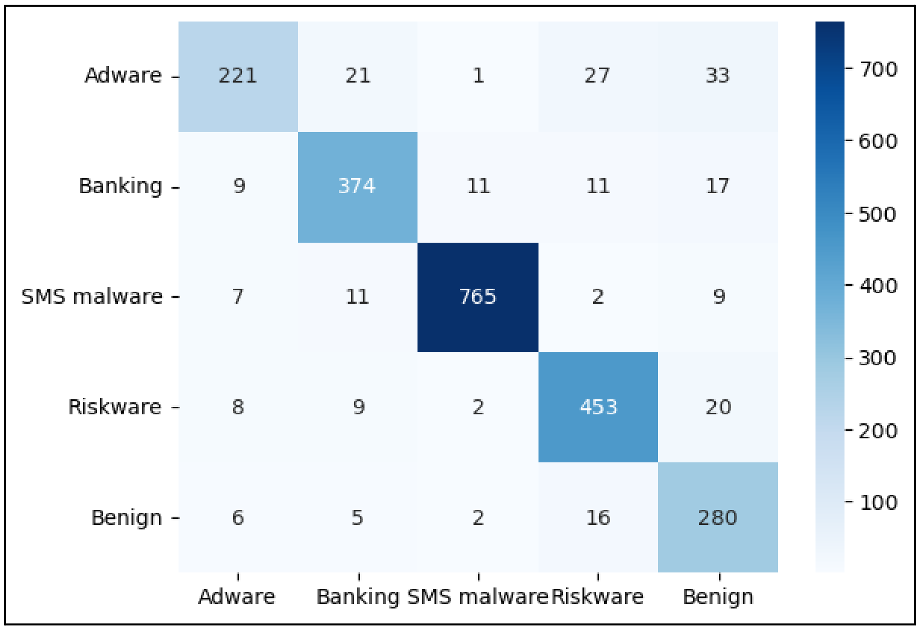
- <u>Random Forest Classification Heatmap</u>
    - Adware Features: 236
    - Banking Features: 374
    - SMS Malware Features: 774
    - Riskware Features: 465
    - Benign Features: 337



- <u>Logistic Regression Classification Heatmap</u>
    - Adware Features: 146
    - Banking Features: 310
    - SMS Malware Features: 762
    - Riskware Features: 363
    - Benign Features: 293



- <u>K-Nearest Neighbors Classification Heatmap</u>
    - Adware Features: 221
    - Banking Features: 374
    - SMS Malware Features: 765
    - Riskware Features: 453
    - Benign Features: 280



	Adware	Banking	SMS malware	Riskware	Benign
Adware	221	21	1	27	33
Banking	9	374	11	11	17
SMS malware	7	11	765	2	9
Riskware	8	9	2	453	20
Benign	6	5	2	16	280

Upcoming Plan: What do you plan to do in upcoming weeks?

- Continue to work on implementing adversarial attacks - try to work on an evasion attack that will attack one of our machine learning models to see how the model performs differently.
- Continue to implement SVM models using FEST and Kernels (Non-linear, linear, and RBF).
- Continue to implement either a Deep Neural Network (DNN) or a Convolutional Neural Network (CNN)
- Incorporate API key given by Androzoo which we were able to gain access to this week.

Obstacles & Concerns**:** Were there any obstacles or barriers that prevented you from getting things done?

- We were originally using Google Colaboratory to work on the code; however, once uploading the data set, Google Drive did not have enough storage for us to work with so we had to transfer over to Visual Studio Code therefore, we had to adjust our collaboration and time management.
- Originally we were looking at using the New Brunswick: CIC-AndMal2017 Dataset however, it seemed to include mostly network traffic. Therefore, we switched to using CICMalDroid 2020 Dataset because it has features related to the permissions, system calls, binders, composite behaviors, sensitive APIs, services, packages, that are extracted from the APKs.
- Clarification:
    - Presentation Date
    - Research Paper Due Date