Policing Android Malware using Object-Oriented Metrics and Machine Learning Techniques

Anand Tirkey[1] [0000−0002−3061−077X], Ramesh Kumar Mohapatra[1] [0000−0002−3424−1465], and Lov Kumar[2] [0000−0002−0123−7822]

[1] National Institute of Technlogy Rourkela, Odisha, India
andy9c@gmail.com, rkmohapatra@ieee.org
[2] BITS Pilani, Hyderabad Campus, Telangana, India
lovkumar505@gmail.com

Abstract. The primary motive of a malware is to compromise and exfiltrate sensitive user data from a system generally designed to uphold the fundamental principles of information security *i.e.*, confidentiality, integrity and availability. Android being the most widely used mobile operating system, is a lucrative ground for malware designers in leveraging system flaws to gain unauthorized user information access. In order to attenuate these issues, it is imperative to design and build robust automated tools for effective android malware prediction. In this paper we bring forward a novel method for android malware detection using object-oriented software metrics and machine learning techniques. 5,774 android apps are collected from Androzoo repository, then it's software metrics are extracted and aggregated using sixteen aggregation-measures which forms the basis of our metrics-based dataset. A total of three hundred and four different machine-learned models are built using various data-sampling techniques, feature-selection methods and machine learning algorithms. Finally, a machine learned model built using SVMSMOTE data-sampling technique applying SPM (Significant Predictor Metrics) feature selection methods over GDCG2H (Conjugate Gradient with Powell/Beale Restarts and 2 Hidden Layers) machine learning algorithm, yields a better malware predictor with AUC (area under ROC curve) value of 0.86.

Keywords: Android Malware Detection · Machine Learning · Object-Oriented Metrics

1 Introduction

Android leads the mobile OS market with a share of 86.1% and is expected to increase to 86.5% by 2021 according to International data corporation (IDC, USA). Meanwhile, in 2019, Google reported that 42.1% of android devices run unsupported versions of the OS. Internet Security Threat Report 2019 (ISTR), published by Symantec shows that it blocked an average of 10,573 malicious mobile apps per day. Malware are more prevalent in apps categorised under Tools (39%), Lifestyle (15%), and Entertainment (7%). It also reported that one in thirty-six mobile devices had high risk apps installed. Karstern Nohl *et al.* [11]

show that security in android ecosystem is further compromised by handset vendors , since they fail to provide timely updates published monthly by Google, to their supported devices. This has created a problem known as Android OS fragmentation for Google, where majority of the mobile devices are devoid of any OS support, which potentially exposes majority of the end-users to malware attacks. Android OS has built-in permission management system, that keeps tab of apps using different permissions, unfortunately the intricacies of this system is too cumbersome for majority of the end-users. Hence, for a malware to gain unwanted access, the end-user just has to ignorantly grant the requested permissions as a matter of habit, without understanding the consequences.

1.1 Objective & Research Questions

The principal objective of this study is to build automated tools towards an effective android malware detection. Another aspect of this study is also to assess the importance of Object-Oriented Software Metrics in evaluating android application packages for malware discovery and mitigation of mobile security risks. The following research questions (RQ) have been put forward in order to identify, analyse and summarize the findings of the experiment proceedings:

- RQ1: Is there an interesting and significant distinction in the performances manifested by the three data sampling techniques ?
- RQ2: Is there a major difference in performance manifested by the three feature selection techniques ?
- RQ3: How do the nineteen classifiers fare in their discriminatory power as adjudged by accuracy and AUC metrics ? Do these classifiers vary greatly in their malware predictive performances ?

2 Related Work

The malware detection methods can broadly be grouped into two categories such as static analysis & dynamic analysis. Many authors have used static analysis such as Zhuo Ma *et al*. [7] use API flows as features for malware detection. They obtain API (Application Program Interface) information from a control flow graph retrieved from an apk, they use this API information to build three API datasets, capturing different aspects of API flows. Neeraj Chavan *et al*. [2] use android permissions requests as their features in malware detection using SVM & ANN. Shivi Garg *et al*. [4] use API and permissions requests as static features & dynamic features such as battery temperature, network traffic etc. Finally, they validate the effectiveness of their model using machine-learning techniques such as SVM, RIDOR, PART, MLP and their ensembles. Yao-Saint Yen *et al*. [13] use apk source code visualization technique. They compute TF-IDF (Term frequency-inverse document frequency) of the decompiled source code and transform that information into images, which is then fed as input to CNN for malware analysis. Alejandro Martín *et al*. [8] have used markov chains and dynamic analysis for malware classification. They have deployed DroidBox tool to gather run-time information and this information is transformed to first-order Markov model. From this model, transition probabilities & state frequencies are used as input data in deep learning algorithm, for malware classification. Dina Saif *et al*. [12] use hybrid set of features retrieved from static-analysis, dynamic-analysis and set of system-calls from both malware & benignware android apps. This hybrid set of features are then used as input for malware classification in Deep-Belief Networks. Ignacio Martín *et al*. [9] have collected android apk malware information through the usage of 61 antivirus softwares. They have then used this information to group android apks into malware classes using graph-community algorithms & hierarchical clustering. Finally, using these groups as their dataset, they have performed malware classification using Logistic Regression & Random Forest Machine Learning Algorithms.

3 Research Methodology

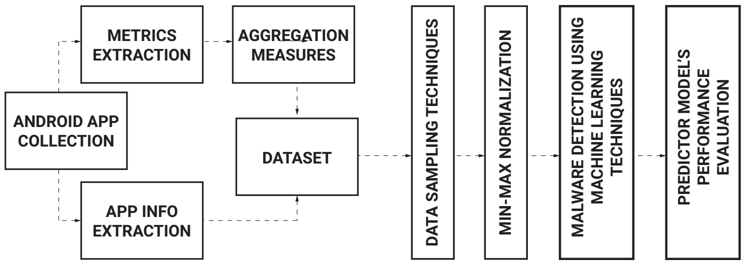

Fig. 1. Malware Detection Model

Table 1. List of Object-Oriented Software Metrics & Aggregation Measures

Efferent Coupling (Ce)	Afferent Coupling (Ca)
Response for Class (RFC)	Weighted Methods per Class (WMC)
Coupling Between Methods (CBM)	Inheritance Coupling (IC)
Number of Children in Tree (NOC)	Average Method Complexity(AMC)
Depth of Inheritance (DIT)	Data Access Metric (DAM)
LCOM3	Lack of Cohesion in Methods (LCOM)
Number of Public Methods (NPM)	Cohesion Among Methods of Class (CAM)
Measure of Functional Abstraction (MFA)	Measure of Aggregation (MOA)
Coupling Between Objects (CBO)	Lines of Code (LOC)

(a) Software Metrics

Hoover index	Variance
Standard deviation	Atkinson index
Maximum	Minimum
Gini index	Kurtosis
First quartile(Q1)	Skewness
Third quartile(Q3)	Median
Theil index	Generalized entropy
Mean	Shannon entropy

(b) Aggregation Measures

3.1 Metrics Extraction and Aggregation Measures

As shown in Figure 1, initially 5,774 android apps are collected from Androzoo [1]. These android packages, are decompiled into JAR (java archive format), for Object-Oriented software metrics extraction using CKJM extended tool [6]. An android app encapsulates multiple classes and each class is represented by a tuple consisting of eighteen Object-Oriented Software Metrics as described in Table 1(a). Therefore, an app is represented as $(n \times 18)$ intermediate matrix, where n represents the total no of classes in an android app. Finally, sixteen aggregation methods as shown in Table 1(b), is applied over this intermediate matrix. Hence, for each aggregation method, an aggregated tuple of eighteen metrics is obtained. Consequently, all the sixteen aggregated tuples are conjoined serially to form a final tuple consisting of $(16 \times 18 = 288)$ metric values. As a result, every android app is finally represented by a tuple with two hundred and eighty-eight features, that forms a tuple of the final metrics-based dataset.

3.2 Data Sampling Techniques

Out of 5,774 android samples, there are 1,582 malwares. It is evident that, there is a disparity in samples for benignware and malware. This class imbalance in a biased dataset affects the overall real-world performance in predicting classes. Therefore, in order to mitigate the class imbalance, three different data sampling techniques are employed such as SMOTE [3] (Synthetic Minority Oversampling Technique), BLSMOTE [5] (Borderline SMOTE) and SVMSMOTE [10] (Support Vector Machine SMOTE). SMOTE uses existing minority classes and interpolates them together to form new minority samples. BLSMOTE uses borderline minority samples in order to generate new synthetic samples, whereas SVMSMOTE employs SVM to detect minority samples which is then used to synthesize new minority samples. Performance of datasets obtained using these data sampling techniques is compared against that of the unsampled original dataset (OD).

3.3 Feature Selection Techniques

As the feature space increases, so does the incurred cost and complexity of building effective machine-learned models. It remains a challenge, to prune irrelevant features without affecting the loss of important information, that ultimately helps in achieving a trade-off between the number of selected features and overall effectiveness of dataset. In this experiment we use three different feature selection techniques other than AM (All Metrics) such as , SPM (Significant Predictor Metrics), ULR (Univariate Logistic Regression) and PCA (Principal Component Analysis). SPM is a set of source-code metrics that are significant predictors of android malware. Initially, t-test is applied over each source-code metric and the metrics with p-values less than 0.05 are considered, that has great discriminatory potential. ULR chooses the best scoring metrics based on various univariate statistical tests and PCA reduces the feature space using Singular Value Decomposition of the data and projects it into a lower dimensional space.

3.4 Classification Techniques

In this experiment, nineteen different classification techniques have been used to create various machine-learned models for malware prediction, such as LOGR (Logistic Regression), DT (Decision Tree), GD(1H/2H/3H) (Gradient Descent with 1/2/3 Hidden Layers), GDM-(1H/2H/3H) (Gradient Descent with Momentum - 1/2/3 Hidden Layers), GDLR-(1H/2H/3H) (Variable Learning Rate Gradient Descent - 1/2/3 Hidden Layers), GDSG-(1H/2H/3H) (Scaled Conjugate Gradient - 1/2/3 Hidden Layers), GDCG-(1H/2H/3H) (Conjugate Gradient with Powell/Beale Restarts - 1/2/3 Hidden Layers), BTE (Best Training Ensemble) and MVE (Majority Voting Ensemble Methods). These techniques are used over various sampled-datasets applying different feature selection techniques. The predictive performance of these datasets are identified and evaluated in order to select a better yielding classifier.

3.5 Performance Evaluation Metrics

The malware prediction potential of a machine-learned model is measured using different standardized evaluation-metrics such as accuracy and error rate. In this experiment, accuracy values and error rate may not reflect the true predictive potential of a machine-learned model, since these evaluation-metrics fail to encompass imbalance and disparity of classes, existing in a dataset. Such imbalanced datasets tend to favour the effective prediction of majority class as compared to the minority class. Therefore, AUC evaluation-metric (area under the ROC curve) is selected as this metric is immune to changes in class distribution in a given dataset. In order to prune any bias between the classes, a 10-fold cross-validation technique is applied while building machine-learned models.

4 Experimental Results & Findings

Based on discussions in Section 3, we formulate and evaluate a null hypothesis H_0: ***"Machine-learned models developed using various data-sampling techniques, feature selection methods, classification algorithms and evaluated using AUC evaluation-metric, for predicting android malware. Indicates no significant performance difference when compared against machine-learned models built using original dataset (OD)"***.

4.1 Analyzing Data Sampling Techniques

In this experiment, three data-sampling techniques are examined as discussed in Section 3.2. Box-plots for OD, SMOTE, BLSMOTE & SVMSMOTE based datasets, depicting accuracy & AUC, along with its descriptive statistics are shown in Figure 2 and Table 2(a), 2(b) respectively. It is observed from Table 2(b) that SMOTE and SVMSMOTE yield a higher median AUC of 0.75. Now considering these four principal metrics-based datasets, a total of $^4C_2 = 6$ unique pairs are possible. Analyzing the p-value of these unique pairs at 0.05 significance level, we can reject a null hypothesis if and only if the p-value is less than $0.05/6 = 0.00833$. In Table 2(c), the p-values less than 0.00833 is denoted by the symbol "$\bullet$". It can be inferred from Table 2(c) that, datasets based on SMOTE and SVMSMOTE are similar between themselves and significantly different from datasets based on OD and BLSMOTE. Table 2(b) shows that SMOTE and SVMSMOTE based datasets yield better AUC median values as compared to OD and BLSMOTE based datasets. Therefore SMOTE and SVMSMOTE based datasets are expected to outperform the rest of the datasets.

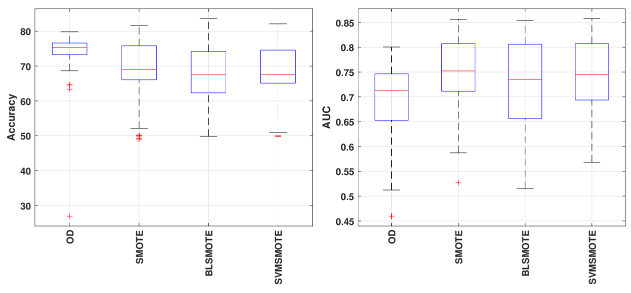

Fig. 2. Box-plots for data-sampling techniques

Table 2. Box-plot descriptive statistics & p-value for data-sampling techniques

	MIN	MAX	MEAN	MEDIAN	Q1	Q3
OD	26.95	79.81	74.17	75.39	73.27	76.6
SMOTE	49.05	81.56	68.91	68.96	66.05	75.8
BLSMOTE	49.82	83.59	67.68	67.48	62.31	74.13
SVMSMOTE	49.76	82.1	68.21	67.56	65.1	74.58

(a) Accuracy

	MIN	MAX	MEAN	MEDIAN	Q1	Q3
OD	0.46	0.8	0.69	0.71	0.65	0.75
SMOTE	0.53	0.86	0.74	**0.75**	0.71	0.81
BLSMOTE	0.52	0.85	0.73	0.74	0.66	0.81
SVMSMOTE	0.57	0.86	0.74	**0.75**	0.69	0.81

(b) AUC

	OD	SMOTE	BLSMOTE	SVMSMOTE
OD		●		●
SMOTE				
BLSMOTE				
SVMSMOTE				

(c) p-values

4.2 Analyzing Feature Selection Techniques

Box-plots for datasets using AM, SPM, ULR & PCA feature selection techniques, depicting accuracy & AUC, along with its descriptive statistics are shown in Figure 3 and Table 3(a), 3(b) respectively. It is observed from Table 3(b) that AM, SPM and ULR based models yield a higher median AUC of 0.75. Now considering these four feature selection techniques, a total of ${}^4C_2 = 6$ unique pairs are possible. Analyzing the p-value of these unique pairs at 0.05 significance level, we can reject a null hypothesis if and only if the p-value is less than $0.05/6 = 0.00833$. In Table 3(c), the p-values less than 0.00833 is denoted by the symbol " $\bullet$ ". It can be inferred from Table 3(c) that, datasets using AM, SPM and ULR are similar amongst themselves and are significantly different from datasets using PCA. Table 3(b) shows that datasets using AM, SPM and ULR yield better AUC median values as compared to datasets using PCA. Therefore datasets applying AM, SPM and ULR are expected to outperform the datasets using PCA.

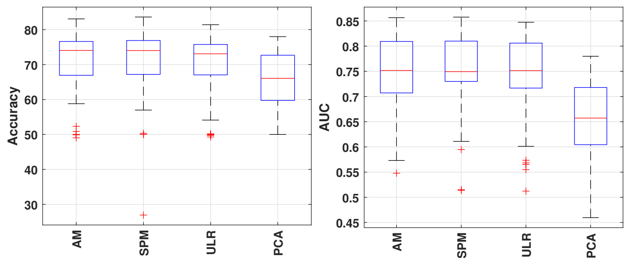

Fig. 3. Box-plots for feature selection techniques

Table 3. Box-plot descriptive statistics & p-value for feature selection techniques

	MIN	MAX	MEAN	MEDIAN	Q1	Q3
AM	49.05	83.05	71.08	74.05	66.92	76.6
SPM	26.95	83.59	71.45	74.01	67.18	76.87
ULR	49.28	81.38	70.72	73.06	67.03	75.74
PCA	50	77.98	65.71	66.05	59.73	72.66

(a) Accuracy

	MIN	MAX	MEAN	MEDIAN	Q1	Q3
AM	0.55	0.86	0.75	**0.75**	0.71	0.81
SPM	0.51	0.86	0.75	**0.75**	0.73	0.81
ULR	0.51	0.85	0.74	**0.75**	0.72	0.81
PCA	0.46	0.78	0.66	0.66	0.6	0.72

(b) AUC

	AM	SPM	ULR	PCA
AM				●
SPM				●
ULR				●
PCA				

(c) p-values

4.3 Analyzing Machine Learning Algorithms

Box-plots for datasets using various machine learning algorithms as described in Section 3.4, depicting accuracy & AUC, along with its descriptive statistics are shown in Figure 4 and Table 4(a), 4(b) respectively. It is observed from Table 4(b) that GDCG2H yields a better median accuracy and AUC of 76.73% and 0.83 respectively. Now considering these nineteen machine learning techniques, a total of ${}^{19}C_2 = 171$ unique pairs are possible. Analyzing the p-value of these unique pairs at 0.05 significance level, we can reject a null hypothesis if and only if the p-value is less than $0.05/171 = 0.00029$. In Table 4(c), the p-values less than 0.00029 is denoted by the symbol ″ • ″. It can be inferred from Table 4(c) that, machine-learned models based on GDCG2H is significantly different from other machine-learned models. Table 4(b) shows that machine-learned models using GDCG2H yields better AUC median values as compared to other machine-learning techniques. Therefore datasets applied over GDCG2H are expected to outperform other classification techniques.

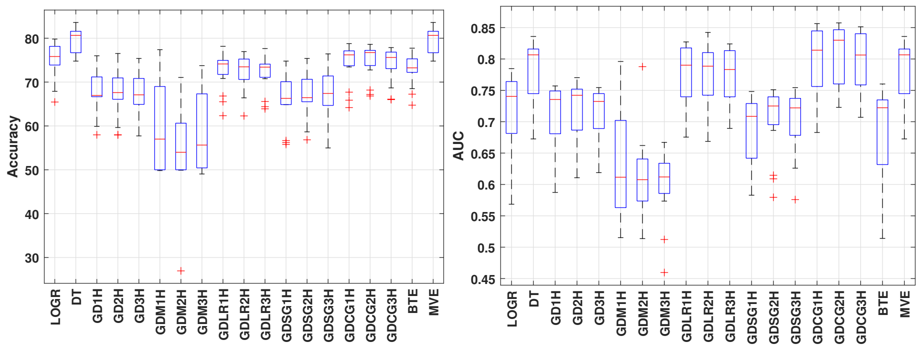

Fig. 4. Box-plots for classifiers

4.4 Analyzing Machine-Learned Models

Upon examining Sections 4.1, 4.2 and 4.3, a total of three hundred and four different machine-learned model's discriminating power are compared using it's accuracy and AUC values. Primarily, a better classifier is characterised by an AUC value closer to 1.0. Consequently, it is expected that a dataset based on either SMOTE or SVMSMOTE applying any of the feature selection techniques like AM, SPM or ULR over GDCG2H classification technique, will yield a better machine-learned model. This expectation is observed and confirmed from Table 5(a) and 5(b), where SVMSMOTE based dataset applying SPM feature selection technique over GDCG2H machine-learning algorithm yields a better accuracy and AUC value of 76.73% and 0.83 respectively. The corresponding ROC curve is illustrated in Figure 5, marked as 2HL.

5 Comparison of Results

RQ1: Is there an interesting and significant distinction in the performances manifested by the three data sampling techniques ? Considering the null hypothesis H_0 and analyzing Section 4.1, it is evident that out of 6 unique pairs,

Table 4. Box-plot descriptive statistics & p-value for classifiers

	MIN	MAX	MEAN	MEDIAN	Q1	Q3
LOGR	65.45	79.81	74.95	75.83	73.89	78.15
DT	74.78	83.59	79.49	80.66	76.69	81.59
GD1H	57.97	76	67.8	66.96	66.66	71.18
GD2H	57.94	76.52	67.6	67.6	66.08	70.95
GD3H	57.73	75.39	67.68	67.09	64.92	70.83
GDM1H	49.82	77.38	59.97	57	50	68.99
GDM2H	26.95	71.06	54.81	54	50	60.63
GDM3H	49.05	73.74	58.73	55.64	50.43	67.32
GDLR1H	62.33	78.16	72.7	74.13	71.75	74.96
GDLR2H	62.29	76.95	72.47	73.48	70.6	75.25
GDLR3H	63.9	77.64	72.1	73.41	71.09	74.1
GDSG1H	55.79	74.78	66.15	66.3	64.87	70.09
GDSG2H	56.84	75.39	67.27	66.44	65.51	70.64
GDSG3H	54.99	76.43	67.54	67.41	64.68	71.4
GDCG1H	64.18	78.77	74.39	76.21	73.71	77.12
GDCG2H	66.83	78.61	74.78	76.73	73.76	77.24
GDCG3H	65.99	77.86	74.15	75.61	73.06	76.84
BTE	64.74	77.73	73.01	73.25	72.23	75.3
MVE	74.78	83.59	79.49	80.66	76.69	81.59

(a) Accuracy

	MIN	MAX	MEAN	MEDIAN	Q1	Q3
LOGR	0.57	0.78	0.72	0.74	0.68	0.76
DT	0.67	0.84	0.78	0.81	0.74	0.82
GD1H	0.59	0.76	0.71	0.74	0.68	0.75
GD2H	0.61	0.77	0.72	0.74	0.69	0.75
GD3H	0.62	0.75	0.71	0.73	0.69	0.74
GDM1H	0.52	0.8	0.63	0.61	0.56	0.7
GDM2H	0.51	0.79	0.61	0.61	0.57	0.64
GDM3H	0.46	0.67	0.6	0.61	0.59	0.63
GDLR1H	0.68	0.83	0.78	0.79	0.74	0.82
GDLR2H	0.67	0.84	0.78	0.79	0.74	0.81
GDLR3H	0.69	0.82	0.77	0.78	0.74	0.81
GDSG1H	0.58	0.75	0.69	0.71	0.64	0.73
GDSG2H	0.58	0.75	0.7	0.73	0.7	0.74
GDSG3H	0.58	0.75	0.7	0.72	0.68	0.74
GDCG1H	0.68	0.86	0.8	0.81	0.76	0.84
GDCG2H	0.72	0.86	0.8	0.83	0.76	0.85
GDCG3H	0.71	0.85	0.8	0.81	0.76	0.84
BTE	0.51	0.76	0.68	0.72	0.63	0.73
MVE	0.67	0.84	0.78	0.81	0.74	0.82

(b) AUC

	LOGR	DT	GD1H	GD2H	GD3H	GDM1H	GDM2H	GDM3H	GDLR1H	GDLR2H	GDLR3H	GDSG1H	GDSG2H	GDSG3H	GDCG1H	GDCG2H	GDCG3H	BTE	MVE
LOGR		●					●	●	●	●	●				●	●	●		●
DT			●	●	●	●	●	●				●	●	●				●	
GD1H							●	●	●	●	●				●	●	●		●
GD2H							●	●	●	●	●	●			●	●	●		●
GD3H							●	●	●	●	●				●	●	●		●
GDM1H									●	●	●				●	●	●		●
GDM2H									●	●	●	●	●	●	●	●	●		●
GDM3H									●	●	●	●	●	●	●	●	●	●	●
GDLR1H												●	●	●				●	
GDLR2H												●	●	●				●	
GDLR3H												●	●	●				●	
GDSG1H															●	●	●		●
GDSG2H															●	●	●		●
GDSG3H															●	●	●		●
GDCG1H																		●	
GDCG2H																		●	
GDCG3H																		●	
BTE																			●
MVE																			

(c) p-values

only two pairs reject the null hypothesis and is marked by the symbol ” • ” in Table 3c. In case, a null hypothesis is rejected, it implies that the distinction identified between samples isn’t by chance and the observation is statistically significant. The machine-learned models based on SMOTE and SVMSMOTE yields better AUC as compared to OD and BLSMOTE. Therefore, SMOTE and SVMSMOTE based models are interesting and manifests significant increase in malware prediction performance as compared against OD and BLSMOTE.

RQ2: Is there a major difference in performance manifested by the three feature selection techniques ? Considering the null hypothesis H_0 and analyzing Section 4.2, it is evident that out of 6 unique pairs, only three pairs reject the

Table 5. Classifier accuracy & AUC against different datasets applying various feature selection techniques

		LOGR	DT	GD1H	GD2H	GD3H	GDM1H	GDM2H	GDM3H	GDLR1H	GDLR2H	GDLR3H	GDSG1H	GDSG2H	GDSG3H	GDCG1H	GDCG2H	GDCG3H	BTE	MVE
	AM	79.46	76.6	74.78	76.52	75.15	64.56	63.38	72.36	78.16	75.11	75.8	72.1	75.39	75.56	77.21	75.72	77.73	77.64	76.6
OD	SPM	79.81	76.6	76	75.13	75.04	77.38	26.95	73.74	76	76.95	75.82	74.09	73.31	76.43	76.95	76.86	76.6	77.21	76.6
	ULR	79.12	76.6	73.92	74.7	75.39	64.64	68.63	71.69	77.3	75.91	77.64	74.78	75.39	75.48	78.77	78.34	76.17	77.73	76.6
	PCA	74.26	74.78	72.96	72.7	72.44	72.44	69.56	72.62	73.4	75.39	73.31	72.42	72.51	72.7	73.98	72.77	74.61	73.22	74.78
	AM	76.79	81.56	66.77	67.88	68.24	73.15	49.91	49.05	75	74.28	74.15	67.1	68.78	67.4	78.16	76.85	77.86	76.01	81.56
SMOTE	SPM	76.25	80.73	69.39	67.24	66.23	67.84	50	62.95	74.7	76.88	73.03	68.08	66.61	67.6	76.61	76.91	77.48	73.27	80.73
	ULR	75.34	80.72	69.15	69.19	69.21	70.15	50.21	49.28	72.67	75.1	72.43	64.84	65.81	67	75.58	77.49	75.06	73.21	80.72
	PCA	67.9	77.15	59.9	57.94	61.58	52.15	53.88	50	66.83	66.41	64.38	56.21	58.65	60.68	65.87	66.83	68.68	68.5	77.15
	AM	76.55	83.05	66.75	68.26	63.6	58.81	50	52.33	74.22	71.18	74.05	64.9	66.35	70.09	77.03	74.76	72.67	73.69	83.05
BLSMOTE	SPM	75.4	83.59	66.57	66.17	67.12	50	50.36	56.98	74.34	72.9	73.93	67.84	65.75	67.94	77.31	77.33	76.79	73.87	83.59
	ULR	73.51	80.61	69.15	67.78	66.89	49.82	54.93	54.3	70.82	73.33	73.87	65.87	65.27	66.75	75.89	74.94	73.45	72.08	80.61
	PCA	65.45	76.78	57.97	58	62.15	53.37	56.12	59.82	62.33	62.29	63.9	55.79	56.84	54.99	64.18	67.18	65.99	64.74	76.78
	AM	78.46	81.61	66.77	67.42	67.06	49.97	50	50.87	74.05	73.63	71.42	65.51	65.99	64.62	73.45	77.15	76.31	74.58	81.61
SVMSMOTE	SPM	77.84	82.1	66.85	67	66.65	50	71.06	61.28	74.91	73.27	73.51	66.73	66.53	67.42	76.52	**78.61**	76.88	72.85	82.1
	ULR	74.58	81.38	67.06	65.99	68.44	50	54.12	49.76	72.91	70.01	70.76	65.45	68.62	64.74	75.06	76.61	73.93	72.37	81.38
	PCA	68.44	77.98	60.84	59.64	57.73	55.19	57.88	52.74	65.51	66.81	65.57	56.66	64.5	61.22	67.7	68.18	66.11	67.24	77.98

(a) Accuracy

		LOGR	DT	GD1H	GD2H	GD3H	GDM1H	GDM2H	GDM3H	GDLR1H	GDLR2H	GDLR3H	GDSG1H	GDSG2H	GDSG3H	GDCG1H	GDCG2H	GDCG3H	BTE	MVE
	AM	0.68	0.72	0.72	0.73	0.72	0.57	0.55	0.67	0.77	0.75	0.76	0.65	0.7	0.7	0.78	0.78	0.79	0.61	0.72
OD	SPM	0.68	0.71	0.73	0.75	0.74	0.74	0.51	0.65	0.76	0.78	0.76	0.71	0.74	0.73	0.8	0.78	0.79	0.59	0.71
	ULR	0.69	0.71	0.74	0.73	0.74	0.57	0.66	0.51	0.77	0.77	0.77	0.71	0.74	0.72	0.8	0.8	0.77	0.62	0.71
	PCA	0.57	0.67	0.59	0.64	0.65	0.52	0.58	0.46	0.7	0.73	0.71	0.63	0.61	0.65	0.72	0.73	0.71	0.51	0.67
	AM	0.77	0.82	0.74	0.74	0.75	0.8	0.66	0.63	0.82	0.81	0.82	0.73	0.75	0.72	0.86	0.85	0.85	0.76	0.82
SMOTE	SPM	0.76	0.81	0.75	0.77	0.75	0.76	0.61	0.66	0.82	0.84	0.79	0.73	0.73	0.74	0.85	0.85	0.84	0.73	0.81
	ULR	0.75	0.81	0.76	0.76	0.75	0.77	0.63	0.64	0.82	0.83	0.79	0.71	0.73	0.74	0.84	0.85	0.84	0.73	0.81
	PCA	0.68	0.77	0.63	0.61	0.64	0.53	0.59	0.59	0.72	0.71	0.69	0.6	0.61	0.63	0.73	0.72	0.74	0.68	0.77
	AM	0.77	0.83	0.73	0.75	0.72	0.64	0.65	0.6	0.83	0.79	0.82	0.7	0.74	0.75	0.85	0.83	0.81	0.74	0.83
BLSMOTE	SPM	0.75	0.84	0.73	0.74	0.73	0.52	0.63	0.61	0.81	0.81	0.82	0.75	0.72	0.74	0.85	0.85	0.85	0.74	0.84
	ULR	0.74	0.81	0.75	0.75	0.73	0.62	0.55	0.62	0.77	0.81	0.81	0.72	0.71	0.73	0.83	0.83	0.82	0.72	0.81
	PCA	0.65	0.77	0.6	0.62	0.66	0.56	0.58	0.61	0.68	0.67	0.69	0.58	0.58	0.58	0.68	0.72	0.73	0.65	0.77
	AM	0.78	0.82	0.74	0.75	0.73	0.66	0.63	0.61	0.81	0.81	0.79	0.7	0.71	0.7	0.8	0.84	0.84	0.75	0.82
SVMSMOTE	SPM	0.78	0.82	0.75	0.75	0.74	0.62	0.79	0.62	0.82	0.81	0.81	0.74	0.74	0.74	0.84	**0.86**	0.84	0.73	0.82
	ULR	0.75	0.81	0.75	0.75	0.75	0.6	0.57	0.57	0.81	0.78	0.78	0.7	0.75	0.72	0.84	0.84	0.81	0.72	0.81
	PCA	0.68	0.78	0.64	0.61	0.62	0.57	0.6	0.58	0.72	0.73	0.72	0.6	0.69	0.66	0.74	0.74	0.73	0.67	0.78

(b) AUC

null hypothesis and is marked by the symbol ” • ” in Table 4c. In case, a null hypothesis is rejected, it implies that the distinction identified between samples isn’t by chance and the observation is statistically significant. The machine-learned models based on AM, SPM and ULR are similar and yields better AUC as compared to PCA. Therefore, AM, SPM and ULR based models are interesting and manifests significant increase in malware prediction performance as compared against PCA.

RQ3: How do the nineteen classifiers fare in their discriminatory power as adjudged by accuracy and AUC metrics ? Do these classifiers vary greatly in their malware predictive performances ? Considering the null hypothesis H_0 and analyzing Section 4.3, it is evident that out of 171 unique pairs, only one hundred and four pairs reject the null hypothesis and is marked by the symbol ” • ” in Table 5c. In case, a null hypothesis is rejected, it implies that the distinction identified between samples isn’t by chance and the observation is statistically significant. Therefore, the machine learning techniques are significantly different amongst themselves and GDCG2H algorithm yields a better AUC of

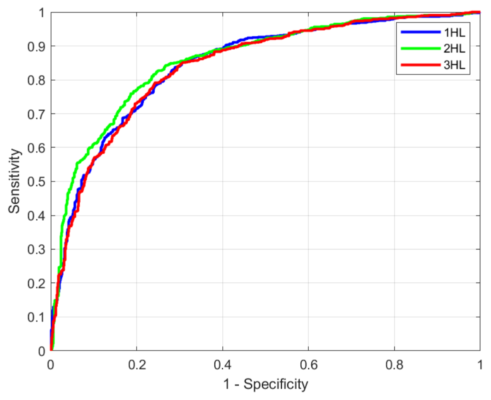

Fig. 5. SVMSMOTE-SPM-GDCG2H classification model's ROC curve

0.86. Hence, machine-learned models based on GDCD2H are interesting and manifests significant increase in malware prediction performance.

6 Threats to Validity

In this work, one possible threat to internal validity identified is that, the machine-learned models built with android apps from a certain point in time may be effective for malwares released within a fixed time-frame (*e.g.* six months or so). And the model may loose it's effectiveness against new strain of malwares released beyond this time-frame. The information regarding this new strain of malwares must trickle down from antivirus companies through Google's VirusTotal service and finally to Androzoo repository. Until, this new information isn't available with Androzoo, it's difficult to build yet another effective machine-learned malware predictor model.

7 Conclusion

Initially android samples are collected over androzoo, which is then decompiled and software-metrics is extracted using CKJM extended tool. For every android app, sixteen different aggregation measures are applied over the extracted eighteen software metrics, which becomes the metrics-based dataset to be used for malware prediction.In order to mitigate benignware and malware sample imbalance in the dataset, three data-sampling techniques are used such as SMOTE, BLSMOTE and SVMSMOTE. In order to reduce the feature space, three feature-selection methods are employed such as SPM, ULR and PCA. Finally, nineteen different classification algorithms are used to build various machine-learned models. A total of three-hundred and four machine-learned models are evaluated using their AUC values. Consequently, machine-learned model built using SVMSMOTE data-sampling applying SPM feature-selection methods over GDCG2H classification algorithm, yields a better AUC of 0.86, which exceeds the malware prediction potential against other models.

References

1. Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y.: Androzoo: Collecting millions of android apps for the research community. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR). pp. 468–471. IEEE (2016)
2. Chavan, N., Di Troia, F., Stamp, M.: A comparative analysis of android malware. arXiv preprint arXiv:1904.00735 (2019)
3. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
4. Garg, S., Baliyan, N.: A novel parallel classifier scheme for vulnerability detection in android. Computers & Electrical Engineering **77**, 12–26 (2019)
5. Han, H., Wang, W.Y., Mao, B.H.: Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing. pp. 878–887. Springer (2005)
6. Jureczko, M., Spinellis, D.: Using Object-Oriented Design Metrics to Predict Software Defects, Monographs of System Dependability, vol. Models and Methodology of System Dependability, pp. 69–81. Oficyna Wydawnicza Politechniki Wroclawskiej, Wroclaw, Poland (2010)
7. Ma, Z., Ge, H., Liu, Y., Zhao, M., Ma, J.: A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE Access **7**, 21235–21245 (2019)
8. Martín, A., Rodríguez-Fernández, V., Camacho, D.: Candyman: Classifying android malware families by modelling dynamic traces with markov chains. Engineering Applications of Artificial Intelligence **74**, 121–133 (2018)
9. Martín, I., Hernández, J.A., de los Santos, S.: Machine-learning based analysis and classification of android malware signatures. Future Generation Computer Systems (2019)
10. Nguyen, H.M., Cooper, E.W., Kamei, K.: Borderline over-sampling for imbalanced data classification. International Journal of Knowledge Engineering and Soft Data Paradigms **3**(1), 4–21 (2011)
11. Nohl, K., Lell, K.: Mind the gap: Uncovering the android patch gap through binary-only patch level analysis. HITB Security Conference (2018)
12. Saif, D., El-Gokhy, S., Sallam, E.: Deep belief networks-based framework for malware detection in android systems. Alexandria engineering journal **57**(4), 4049–4057 (2018)
13. Yen, Y.S., Sun, H.M.: An android mutation malware detection based on deep learning using visualization of importance from codes. Microelectronics Reliability **93**, 109–114 (2019)