PERMISSION BASED MALWARE DETECTION ANALYSIS IN ANDROID APPLICATIONS

UĞUR PEHLİVAN

SEPTEMBER 2014

PERMISSION BASED MALWARE DETECTION ANALYSIS IN ANDROID APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UĞUR PEHLİVAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2014

PERMISSION BASED MALWARE DETECTION ANALYSIS IN ANDROID APPLICATIONS

Submitted by **Uğur PEHLİVAN** in partial fulfillment of the requirements for the degree of **Master of Science in the Department of Information Systems**,
Middle East Technical University by,

Prof. Dr. Nazife Baykal ____________________
Director, Informatics Institute

Prof. Dr. Yasemin Yardımcı Çetin ____________________
Head of Department, Information Systems

Prof. Dr. Nazife Baykal ____________________
Supervisor, Information Systems, METU

Examining Committee Members

Assoc. Prof. Dr. Aysu Betin Can ____________________
Information Systems, METU

Prof. Dr. Nazife Baykal ____________________
Information Systems, METU

Assist. Prof. Dr. Aybar Can Acar ____________________
Medical Informatics, METU

Dr. Serkan Alkan ____________________
Medical Center, METU

Assist. Prof. Dr. Tuğba Taşkaya Temizel ____________________
Information Systems, METU

Date: 02.09.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Uğur Pehlivan

Signature:

ABSTRACT

PERMISSION BASED MALWARE DETECTION ANALYSIS IN ANDROID APPLICATIONS

Pehlivan, Uğur
M.S., Department of Information Systems
Supervisor: Prof. Dr. Nazife BAYKAL

September 2014, 72 pages

Android mobile devices have developed very fast in past decade and have been very widespread in all over the world. Nowadays, several applications are available on application markets. The number of android applications also increases with the increase in the variety of applications. Those applications may become very dangerous for the users of android mobile devices because of fast development and wide variety of applications. Some applications may have the malicious activities such as novelty and amusement, selling user information and stealing user credentials etc. For this reason, the detection of malicious android applications has become very important in recent years for the security of mobile device's users. In this study, the permissions required for the installation and running processes of android applications were analyzed to determine best performing feature selection methods and classification algorithms which are used for detecting the malicious applications in android mobile devices. 4 feature selection methods consisted of attribute based and subset based selection methods used to reduce the number of attributes and to increase the performance of classification algorithms. The classification algorithms were chosen from the Bayesian, decision tree and SVM classification algorithms in order to compare the performance of different type of classification algorithms. Moreover, the effect of dataset size was investigated to measure the performance of classification algorithms. The permissions are also analyzed in accordance with their presence in the malicious applications by using the clustering analysis.

Keywords: static analysis, feature selection, classification, clustering, malware detection in android applications

ÖZ

ANDROİD UYGULAMALARDA KÖTÜ AMAÇLI YAZILIMLARIN TESPİT EDİLMESİNE YÖNELİK ANALİZ

Pehlivan, Uğur
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Prof. Dr. Nazife Baykal

Eylül 2014, 72 sayfa

Android mobil cihazlar son yıllarda çok hızlı gelişerek tüm dünyada yaygın bir şekilde kullanılmaya başlanmıştır. Mobil cihazların kullanımındaki artış ile birlikte mobil uygulamalarda hızlı gelişim göstermiş olup uygulama marketlerinde çok çeşitli uygulamalar kullanıcılara sunulmaktadır. Mobil cihazların ve bu cihazlarda kullanılan uygulamalardaki hızlı gelişim, bu cihazlardaki güvenlik ihtiyacının ortaya çıkmasına neden olmuştur. Uygulama marketlerinde sunulan bazı uygulamalar kötü niyetli olabilmekte olup kullanıcılar için güvenlik tehdidi oluşturabilmektedir. Bu kötü niyetli uygulamalar, kullanıcıların özel bilgilerinin çalınmasına ya da cihazlarının kötü amaçlı kullanılmasına neden olabilmektedir. Bu nedenle android uygulamalarda kötü amaçlı uygulamaların tespit edilmesi son yıllarda önem kazanmıştır. Bu çalışmada android uygulamalara cihaza kurulumu için verilmesi gereken izin bilgileri analiz edilerek kötü amaçlı uygulamaların tespit edilmesi amaçlanmıştır. Analizlerde özellik seçim metotları ve sınıflandırma algoritmalarının performansları değerlendirilerek en iyi performans gösteren metot ve algoritmalar belirlenmiştir. Özellik seçme yöntemleri, özellik bazlı ve küme bazlı olmak üzere iki türlü seçim yönteminden oluşan 4 özellik seçme yönteminden oluşmaktadır. Özellik seçme yöntemleri kullanılarak sınıflandırma algoritmalarının performansının arrtırılması amaçlanmıştır. Sınıflandırma algoritmaları ise bayesian, decision tree ve Support Vector Machine olmak üzere üç çeşit algoritmadan seçilen 5 sınıflandırma algoritmasından oluşmaktadır. Ayrıca, veri setinin büyüklüğünün sınıflandırma algoritmalarının performansı üzerindeki etkisi bu çalışmada incelenmiştir. Uygulama izinlerinin kötü amaçlı uygulamalarda bulunma karakterleri kümeleme yöntemi kullanılarak analiz edilmiştir.

Anahtar Kelimeler: statik analiz, özellik seçimi, sınıflandırma, kümeleme, android uygulamalarda kötü amaçlı yazılımı tespit etme

This thesis is dedicated to:

To My Parents

ACKNOWLEDGMENTS

I want to thank to my Thesis supervisor Prof. Dr. Nazife Baykal for her valuable advices and mentoring during my MSc study. I really appreciate her efforts to make me familiar with IS research.

I am very thankful to my family for their continuous support during my whole life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

APK: Android Package Kit

APN: Access Point Name

CART: Classification and Regression Tree

FNR: False Negative Rate

FPR: False Positive Rate

IEEE: Institute of Electrical and Electronical Engineers

IS: Information Systems

Malware: Malicious Software

MMS: Multi-Media Messaging Service

OS: Operating System

RF: Random Forest

SMS: Short Message Service

TNR: True Negative Rate

TPR: True Positive Rate

CHAPTER I

1 INTRODUCTION

1.1 Introduction to Mobile Devices and Android Applications

The popularity of mobile devices has shown sharp increase in their usage by the people in last decade because of the developments made in the mobile devices. The capabilities and functionalities of mobile devices increase very fast which attracts people for using those mobile devices. Mobile devices have become nearly our desktop computers on hand since mobile devices have most of the functionalities of desktop computers. A closer look at the purpose of using mobile phones reveal that they are mostly used for web browsing, social networking, and online banking. In addition, they are used for mobile-specific functions such as SMS messaging, continuous updating of location data, and ubiquitous access (Felt, et al., 2011). As the capabilities in mobile phone functionality increase, mobile phones become more and more attractive for a wider population. Market data surveys reveal that the number of smart phone sales worldwide reached a record of about 208 millions in 2012. This was a 38.3% increase compared to the sales in the previous year (Suarez-Tangil, et al., 2014). Mobile phone applications have found a rich variety of applications not only in common daily life activities but also in users with specific needs. From games to multimedia applications, navigation systems, and health-related applications, they are recently available in mobile application markets, such as Google Play Market and Apple Store. All the markets have been growing steadily both in terms of the applications offered to the users and downloads from markets performed by the users (Statista, 2014). The applications downloaded from Google Play have been increasing faster in every year (The number of cumulative app downloads from the Google Play app store for Android devices between August 2010 to July 2013, 2014). The fast increase in popularity and functionality of smart phones has also increased their potential as a target for malicious activities, since users access various information content by means of mobile applications (Mobile Security Guide: Protect Your Organization From Mobile Malware, 2013). Some of applications downloaded from those application platforms are injected and show malicious activities when they are installed on the mobile devices. Apple store makes regular checks and controls on the applications listed in their application platform by applying a policy that is subjected to strict registration and company-issued digital certification before the release of any application regularly. The malware applications are rarely encountered in Apple store. However, Google Play does not have a controlling mechanism against malicious applications in their application market. If the users return feedback as an application has malicious activity then Google Play may get rid of that application from application market. For this reason, the detection of malware applications is very significant especially in android applications.

Malware detection analysis of android applications may be performed in two ways; static analysis and dynamic analysis (Amamra, et al., 2012). Static analysis is the analysis which is made without running the android application by using some of the properties extracted from application code of android applications. Dynamic analysis Amos (2013) is realized after running the android application by monitoring run-time behaviors of android application. Dynamic analysis is more difficult than the static analysis because of the time and effort requirements of running the android application and monitoring the changes on some parameters occurred in the mobile device such as battery consumption (Thomas, et al., 2013). In this paper, static analysis is used in the analysis by using the permission data gathered from the application data of android applications.

Android applications are working by using some functions available on the mobile devices. In order to use required functions, related permission about using that function must be given by the mobile device, actually by the user of mobile device while installing the application. For this reason, mobile device asks for the permissions required by that application before installing the application on mobile device. If the user accepts the permissions asked by the application; the application is then installed on the mobile device. The permissions required by the application determine the capability of installed application on the mobile device. Hence, those permissions required by the mobile device applications are valuable and worth to be investigated to detect malicious applications. This type of investigation is classified as static analysis because the analyzer does not run the program and only searches the effects of those permissions on the maliciousness of applications. The permissions required by the android applications are used in this paper as a data. The permission list given in Developers (2014) shows possible permissions presented in API Level 20 that can be asked by the android applications while installing on the mobile device.

Feature selection methods are used to reduce the dataset size by removing some of the features (attributes) which are not useful to be used in the analysis. The remaining features are selected by considering the representation capability of whole feature in the dataset. Efficient feature selection methods introduce performance gains in classification analysis, therefore it is usually considered as a necessary step for preparing a dataset for classification. Some advantages of feature selection methods are reducing the dataset size, decreasing the workload spent on the dataset and reducing the time of analysis. For this reason, using a feature selection method is an important part of preparing dataset for the analysis. Choosing an appropriate feature selection method is getting more important concept in the analysis of dataset which is seriously affecting the analysis phase of the study. For this reason, the performance of feature selection methods with different classification algorithms that are shown in table 1.1 is investigated in this study. Two attribute based; Gain Ratio Attribute, ReliefF Attribute and two subset based; Cfs Subset, Consistency Subset feature selection methods are used in the preparation of datasets. In addition, 5 classification algorithms are chosen from Bayesian, Decision Trees; Classification and Regression Tree, J48 Decision Tree and Random Forest and SVM; Sequential Minimal Optimization (SMO) algorithms.

Table 1.1: Feature selection methods and classification algorithms for data analysis

Feature Selection Methods	Classification Algorithms
• *Gain Ratio Attribute* • *ReliefF Attribute* • *Cfs Subset* • *Consistency Subset*	• *Bayesian Classification* • *Classification and Regression Tree (CART)* • *J48 Decision Tree* • *Random Forest* • *Sequential Minimal Optimization (SMO)*

1.2 Study Objectives and Research Questions

The primary objective of this study is to investigate the effect of permissions on the detection of malware applications in android platforms. The contribution of permission data within the application code to the effectiveness of permission based malware detection in android applications is investigated. Since those permissions are asked when installed on the mobile devices to use some resources of mobile device. Secondary objective is to measure the performance of feature selection methods and classification algorithms on permission based malware detection. In addition, the contribution of clustering analysis are analyzed and evaluated to determine which permissions are determinant in malicious applications. Another objective of this research is to identify the effect of dataset size on the accuracy of classification algorithms for malware detection in android platforms. The performance comparison is realized by using datasets including different number of instances. The following research questions are tried to be answered in this study:

1. Which classification algorithm show better performance with higher accuracy for the detection of malware applications among Bayesian, Classification and Regression Tree (CART), J48 Decision Tree, Random Forest and Sequential Minimal Optimization (SMO)?

2. Which classification algorithm, feature selection method and the number of selected features combination is more accurate in detection of malware in android applications?

3. Is clustering algorithm useful on determination of which permissions have more tendencies to be used by malicious applications?

4. Is the dataset size effective on the performance of classification algorithms?

1.3 Scope

This study is introducing the analysis of permission data for the detection of malicious applications in android applications by using well known methods and algorithms which are widely used in the literature. 3,784 android applications available on the Google Play Market are made use of in this study. 4 feature selection methods, 5 classification algorithms and 1 clustering algorithm are implemented within the analysis and evaluated by using permissions data of those 3,784 android applications.

1.4 Significance of the Study

The detection of malware applications is very important nowadays because of widely usage of mobile devices throughout the world. Those mobile devices are holding very important information of people. Malicious applications that can be installed on the mobile device may threaten the people lives causing the tangible or intangible loss. For this reason, the researchers are concentrating on the detection of malware in mobile security.

In the literature, there are studies made about detection of malicious applications. Those studies are realized by using static or dynamic analysis methods. Static analysis about malware detection in mobile applications is mostly available in the literature. However, the studies about dynamic analysis are rarely available because of the difficulty of dynamic analysis like time requirements and special platforms for monitoring mobile device. Studies made by static analysis are using application data which are existed within the code of application or data presented in application markets. The data is analyzed by using classification algorithms to perform the detection of malicious applications. Feature selection methods are used in some studies to increase the performance of the classification algorithms. After implementing feature selection method and classification algorithms, the results of classification are evaluated in the studies. In addition, clustering analysis is realized to analyze the properties of malicious applications. The comparison of some of the studies published in the literature and this study presented in table 1.2.

An effective approach was presented to detect malware in mobile applications based on Bayesian classification models obtained from static code analysis in Yerima et al. (2013). The models were built from a collection of code and application characteristics providing potential malicious activities. They used API call detectors, command detectors and permission detectors as a data by reverse engineering the applications in their studies. They studied on different sets of features were used containing 5, 10, 15 and 20 features. They used mutual information for feature selection to increase the performance of their classification. The evaluation of datasets having different feature set was performed in this study.

Abu Samra et al. (2013) made clustering analysis by using the permissions of mobile devices. K-means clustering algorithm is made use of to divide dataset into clusters. They used 18,174 mobile applications to cluster into two categories non-malicious applications and malicious applications. They evaluate the results of analysis with precision, recall and F-measure.

Table 1.2: The comparison of studies in literature with this study

Studies	Detection Method	Data	Feature Selection Method	Classification Algorithm	Clustering Algorithm	Dataset Size
Yerima et al. (2013)	Static Analysis	API Call Detectors Command Detectors Permissions Detectors	Mutual Information (MI)	Bayesian Classification	No Clustering	2,000 APK Files (1,000 Benign-1,000 Malware) Totally, 58 Features 5, 10, 15, 20 features
Abu Samra et al. (2013)	Static Analysis	Permissions defined in AndroidManifest.xml	No Feature Selection	No Classification	K-Means Clustering Algorithm	18,174 APK Files
Amos at al. (2013)	Dynamic Analysis	Run-time data	No Feature Selection	Random Forest Naïve Bayes Multilayer Perceptron Bayes net Logistic J48	No Clustering	1,777 Applications 30 features
Zaw & Aung (2013)	Static Analysis	Permissions defined in AndroidManifest.xml	Information Gain	J48 Random Forests CART	K-Means Clustering Algorithm	200 APK Files, 160 Features 500 APK Files, 160 Features
Thomas et al. (2013)	Dynamic Analysis	Power Consumption	No Feature Selection	Power Consumption Histograms MFC Coefficients and Gaussian Mixture Models	No Clustering	96 captured measurements 247 data points for each measurement
In This Study	Static Analysis	Permissions defined in API Level 20	Attribute based Gain Ratio Attribute ReliefF Attribute Subset based Cfs Subset Consistency Subset	Bayesian Decision Trees; CART J48 Random Forest SVM; Sequential Minimal Optimization (SMO)	K-Means Clustering Analyzing the feature properties in clusters	3,784 APK Files Different dataset sizes; 500, 1,000, 2,500 and 3,784 instances Different feature numbers; 25 and 50 features selected

Permission data extracted from manifest file is used in classification analysis in Zaw & Aung (2013). Information gain enables in this study to increase the performance of classification algorithms. In this study, both classification and clustering algorithms are included in their methodology. The decision tree algorithms namely; CART, J48 and Random forest are used for the performance comparison of algorithms. The dataset size in this study is very small with 200 and 500 instances.

Dynamic analysis is realized in Amos et al. (2013) and Thomas et al. (2013) by using run-time data. Power consumption is observed in Thomas et al. (2013).
The dataset used in this study to answer research questions stated in this study is taken from COMODO laboratory. The dataset contains 3,784 mobile applications with their status whether they are benign or malware. COMODO laboratory is

working on the mobile security to provide secure usage of mobile devices for their customers. The application codes are manually investigated line by line by the analyzers to determine whether the application involves malicious activities or not. The reason of signing applications as malware is carefully kept in their databases. For this reason, the status of applications used in this study is thought as reliable. The dataset includes 2,338 benign applications and 1,446 malicious applications. The data used in this study is extracted from the application code by searching the Androidmanifest.xml files. The permissions published as API level 20 by Android Developers (2014) is searched within the Androidmanifest.xml file. API level 20 is the latest published permission list by Android. API level 20 consists of 180 permissions. The number of permissions used in previous studies was small because the number of permissions is increasing continuously; 20 API levels have been presented by android until now. In the future, the number and content of permissions will expand naturally as mobile devices and android applications continue to develop.

Three types of classification algorithms are utilized in this study namely; Bayesian, decision tree and support vector machine as shown in table 1.2. Classification and Regression Tree (CART), J48 decision tree algorithm and Random Forest algorithm are chosen from the decision tree classification algorithms. Sequential minimal optimization (SMO) classification algorithm is selected from as support vector machine algorithm. In this study, the classification algorithms from different classification types are intended to be compared. Decision tree algorithms; CART, J48 and Random forests were also used in Zaw & Aung (2013) in order to measure their performance. Only Bayesian classification algorithm is made use of in Yerima & Sezer (2013). SMO classification algorithm has not been seen in the literature for usage of malware detection in android applications. This study contributes to the literature by using three types of classification algorithms namely; Bayesian, decision trees and SVM. By this way, the best performing classification type is tried to be found in this study. One of the aims of this study is to implement and compare 5 classification algorithms on same dataset to better evaluate their performances. In previous studies, those classification algorithms were not used on same dataset especially SVM. In this way, first research question is tried to be answered by making performance comparison of those classification algorithms with same dataset.

The feature selection methods are selected from attribute based and subset based feature selection methods in order to compare attribute based and subset based methods. For this reason, two of those methods are chosen from attribute based feature selection methods and the other two methods are chosen from subset based feature selection methods. The feature selection methods are Gain Ratio Attribute, ReliefF Attribute, Cfs Subset and Consistency Subset feature selection methods as given in table 1.2. In the literature, gain ratio attribute and information gain attribute feature selection methods are found in studies related with malware detection in mobile applications. Only a feature selection method is chosen and implemented in the previous studies in the literature without looking at which feature selection method is more convenient to classification algorithm used in the studies. In some of the studies, the feature selection was not implemented. There is no study for performance comparison of feature selection methods in malware detection. ReliefF attribute, Cfs subset and consistency subset feature selection methods might not be used before in malware detection studies until now. The comparison of those feature selection methods is realized in this study to answer second research question. The feature numbers are chosen 25 and 50 to determine which best represents the all features in the dataset.

The last research question that is tried to be answered in this study is the performance of classification algorithms with the changing dataset sizes. Whether the performance of classification algorithm increases or not with the larger datasets is an important question. The analyzers working on the large dataset sizes may spend their time for nothing. In literature, there is one study questioning the dataset sizes by working on only two datasets having 200 and 500 instances. (Zaw & Aung, 2013) In this study, four datasets are used to measure the classification performances. In addition, the dataset sizes are much larger than the datasets used in Zaw & Aung (2013) because the dataset sizes changes from 500 to 3,784 instances in this study. The optimum dataset size is also questioned in this study to give a clue to analyzers working in this field. This part of the study will contribute to the literature in a way that the results of this study will help the researchers in selecting their dataset sizes.

The performance of clustering algorithm is questioned in this study as well. K-means clustering algorithm is used in the implementation of clustering analysis. Since K-means clustering algorithm is simple and easy way to classify a dataset through a given number of clusters. (Abu Samra, et al., 2013) K-means clustering algorithm is used in the literature but the results of the clustering algorithm is not analyzed to see feature properties. The aim of clustering analysis in this study is to analyze the features according to the cluster characteristics. The status of the data is known that is they are available within the dataset. This status is removed from the dataset in order to measure the performance of clustering analysis in the detection capability of this status value. After implementing the clustering analysis, the cluster characteristics are investigated with different perspectives. In addition, the features which are strong capability for the detection of malware is tried to be determined by using clustering results. Those features might be used for further investigation in the literature.

To sum up, some of the research questions of this study were worked in different studies in the literature. However, the datasets used for those analyses are different. In this study, all those analysis are realized on the same dataset in order to make it possible a reliable comparison of results. Other studies in the literature made comparisons between their results and other results however the dataset used in the studies are different. For this reason, the comparison becomes inadequate and poor because of the difference in datasets. This study contributes the literature with implementing all those analysis explained above on the same dataset. The dataset size is also studied to measure the effect on the result of feature selection methods and classification algorithms. In addition, the classification algorithms Bayesian, CART, J48, Random Forest and SMO used together first time in this study. Bayesian and decision tree algorithms were not used in the same study before. Moreover, SMO classification algorithm was not used for malware detection purpose before. ReliefF attribute, Cfs subset, Consistency subset feature selection methods are also firstly made use of in this study. The compatibility of four feature selection methods with classification algorithms are questioned in this study as well. There is no previous study questioning the suitability of classification algorithms and feature selection methods. The clustering analysis made in this study which is not much studied compared with classification analysis in the literature. (Abu Samra, et al., 2013) This study will contribute by using both classification algorithms and clustering algorithms in the same study as well.

1.5 Structure of the Research

Figure 1.1 illustrates the research progression from Literature Review to Conclusion in this study.

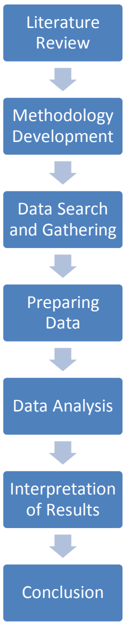


Figure 1.1: Study progression

1.6 Overview of Contents

This study contains five chapters namely; Introduction, Literature Review, Research Methodology, Results and Evaluation, and Conclusion and Implications.

Chapter-1 includes a brief introduction to the mobile devices and android applications. It also includes study objectives, research questions, scope of the thesis, significance of study and finally, the overall structure of the research.

Chapter-2 reviews the previous research studies regarding the thesis topic. The main information about mobile devices, android applications, malware, android system architecture and malware detection methods are discussed in this chapter.

Chapter-3 contains the data preparation processes from data collection to the prepared data which is used in the analysis. Moreover, the details of analysis made in this study are explained in this chapter.

Chapter-4 includes the implementation of analysis and the evaluation of the results obtained in those implementations.

The discussion of the analysis results, conclusions and implications for future studies are given in Chapter-5.

CHAPTER II

2 LITERATURE REVIEW

A comprehensive literature review is realized at the beginning of this study to overview past studies made related with topic of this study. At the beginning of literature view, “Malware”, “Malware classification”, “Malware clustering”, “Malware detection” keywords are used to find related studies about malware classification and clustering topic. A research pool is constructed as a result of made research by those keywords. The studies found according to the first searching consists of the studies about classification or clustering analysis in both desktop platform and mobile platform but they are mostly the studies about the desktop platforms. According to the gains taken from the first stage of searching, it is decided that the searching criteria should be narrowed down to the mobile platforms because the studies in the mobile platforms are more insufficient when compared with the desktop platforms. The reason of this is the level of development in the desktop platforms against the mobile platforms. The background of desktop platforms relies on two and three decades before. Mobile platforms have been developed very fast in past decade which made mobile platforms as popular as desktop platforms. To direct literature view to in the field of mobile platforms, the keywords are revised by adding “mobile” word to the previous used keywords. The keywords used to search for related works are “Mobile malware”, “Android applications”, “Mobile Malware classification”, “Mobile Malware clustering” and “Mobile malware detection”.

2.1 Mobile Devices and Android Applications

The popularity of mobile devices has shown sharp increase in their usage by the people because of the developments made in the mobile devices. According to Gartner (2014), worldwide sales of smart phones to end users totaled 968 million units in 2013, an increase of 42.3 percent from 2012.

In Thompson (2012), it is stated that “there are currently around 675,000 applications in the official Google’s Android market; with an estimated 25 billion downloads (as at October 2012)”. In the smart phone OS market, Android's share grew 12 percentage points to reach 78.4 percent in 2013 according to Gartner (2014). The applications downloaded from Google Play have been increasing faster in every year. (The number of cumulative app downloads from the Google Play app store for Android devices between August 2010 to 2013, 2014).

The writer of malicious applications has some benefits from undertaking the risk of this illegal activity. The incentives for writing mobile malware can be listed as following; (Mohata, et al., 2013)

- *Novelty and Amusement:* This type of malware causes to mischief or damage to amuse the author.

- *Selling User Information:* Large amounts of users' information exist in the mobile phones. This type of malware steals the information of user and sells others.

- *Stealing User Credentials:* Credentials like passwords and payment information are the important information about the users of the mobile phone. Those credentials could be used by the malware author to take financial gain.

- *Premium-Rate Calls and SMS:* This type of malware uses any mobile phone as their own mobile by using their calling or SMS services especially premium rate calls and SMS which are very costly for the mobile phone user. This type of malware may not be detected until the next bill of the mobile phone user.

- *SMS Spam:* SMS spam is used for commercial advertising and spreading phishing links. The victim or compromised mobile phone is used as a tool for sending this type of SMS by the malware authors.

- *Search Engine Optimization:* This type of malware sends the requests to the search engine for a specific search term to increase the rate of term in the search engine. In this way, a specific website link is coming at the beginning of search results. The victim mobile phone is used for making those search requests to the search engine many times.

- *Ransom:* Any information on the mobile phone that is stolen by malware or any condition created by malware that distracts the user of mobile phone can be used for blackmail. Money or some other thing could be demanded by the author of malware from the user of mobile phone in order to get rid of blackmail.

Mobile devices having android operating system has been increasing very much when compared with the other operating systems in mobile devices as shown in figure 2.1. This explains why the android mobile device users are main target of the above damages because it becomes cheaper for per targeted user than any other operating system for the writers of malicious applications.

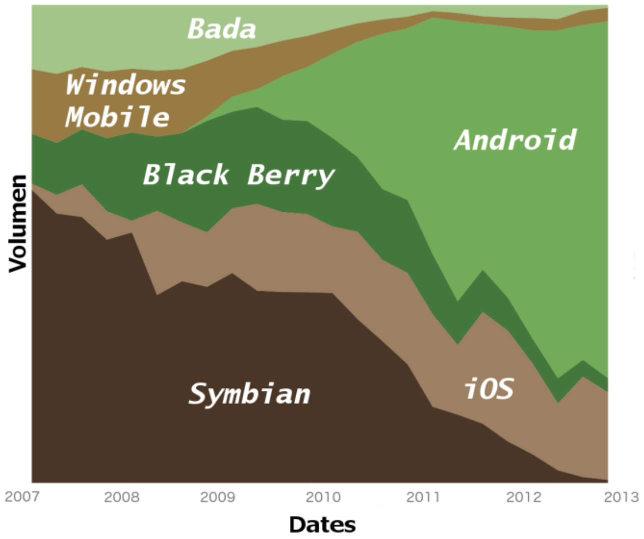


Figure 2.1: Main smart phone platforms by market share from 2007 to 2012 (Suarez-Tangil, et al., 2014)

2.2 Android System Architecture

Android platforms have layered system architecture as it is shown in Figure 2.2. The architecture of android system includes User applications, application framework, libraries, android runtime and linux kernel drivers. Yerima et al. (2013) states that Android is effectively a software stack for mobile devices that includes an operating system, middleware and key applications and uses a modified version of the Linux kernel. The libraries on the linux kernel provide most of the functionalities of android system. The application framework layer provides all API's for accessing the device hardware. At top of operating system, the user applications are located which are installed from application market.

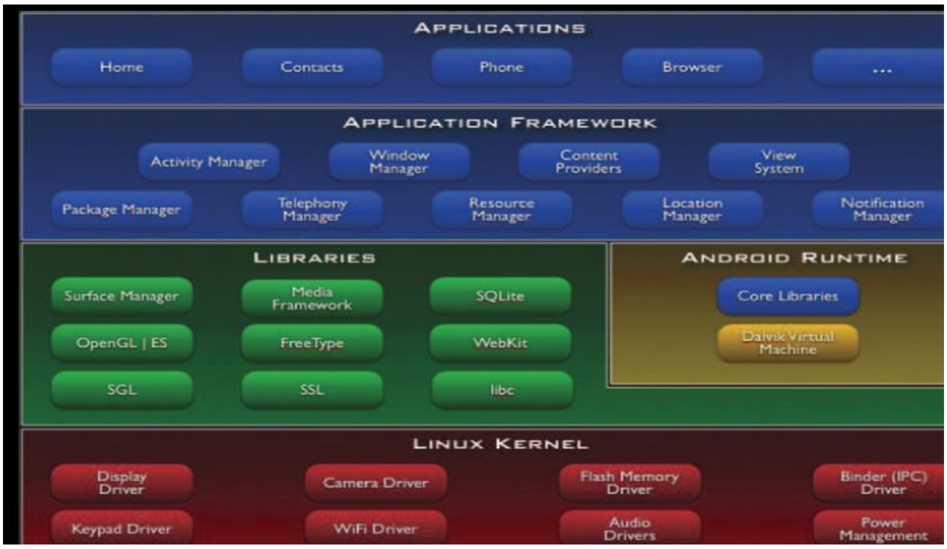


Figure 2.2: Android system architecture (Yerima, et al., 2013)

2.3 Malware Detection Methods

In general, the analysis in the detection of malware divided into two groups mainly; static analysis and dynamic analysis. Static analysis is the analysis made by using the data existed before running the application. Dynamic analysis is the analysis made by using the data obtained after running the application. Hence, the main difference between static and dynamic analysis is where to obtain the data used in the analysis whether it is run-time data or not.

Amamra et al. (2012) presents a good and detailed classification of smart phone malware detection techniques. Figure 2.3 gives the general view of their classification of malware detection techniques. They divided the detection techniques according to the detection way of the technique. Signature-based detection technique models the known malicious behavior of malware in the form of signature and uses this signature in the detection phase. But, in anomaly-based detection technique, there are two phases which are training phase and detection phase. According to the their study, system normal behavior profile is determined in the training phase and the deviation in the detection phase that exceeds a predetermined amount this normal behavior profile is considered as anomaly.

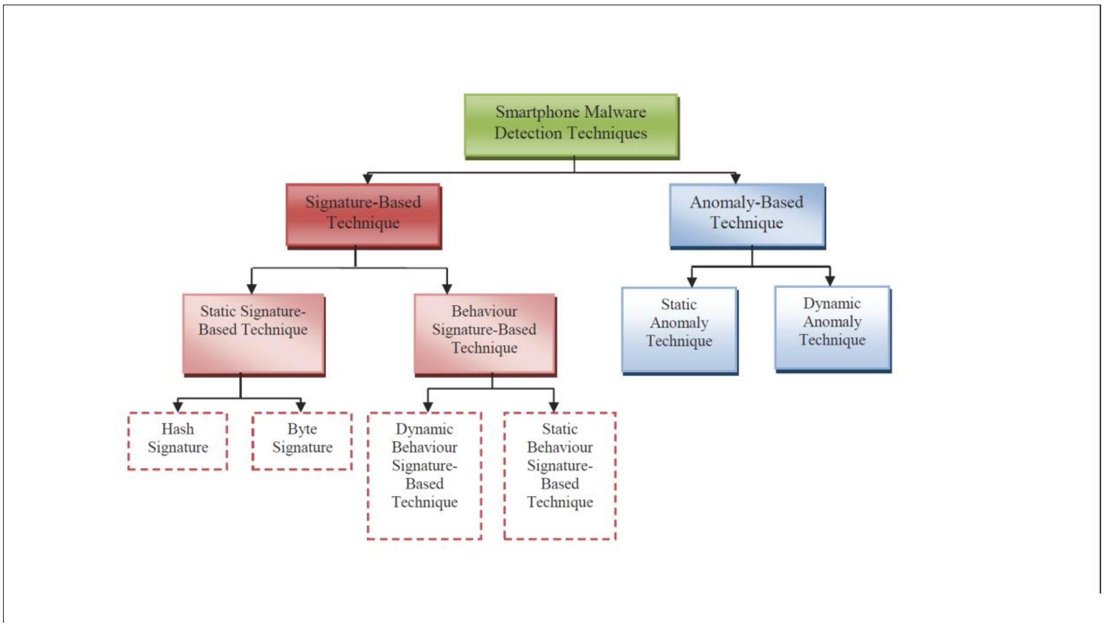
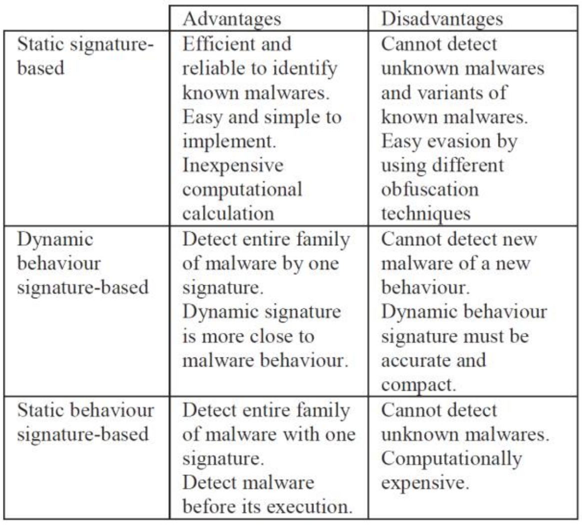

Smartphone Malware Detection Techniques
Signature-Based Technique
Anomaly-Based Technique
Static Signature-Based Technique
Behaviour Signature-Based Technique
Static Anomaly Technique
Dynamic Anomaly Technique
Hash Signature
Byte Signature
Dynamic Behaviour Signature-Based Technique
Static Behaviour Signature-Based Technique

Figure 2.3: A classification of smartphone malware detection techniques (Amamra, et al., 2012)

Amamra et al. (2012) classified the behavior signature based technique into static behavior signature and dynamic behavior signature techniques. The signature is extracted by analyzing the application code in the static behavior code. However, the signature is extracted from runtime information by monitoring the application in the dynamic behavior signature based technique.

The advantages and disadvantages of static signature based, dynamic behavior signature-based and static behavior signature-based techniques are also provided in Amamra et al. (2012) which is shown in table 2.1.

Table 2.1: Advantages and disadvantages of signature-based detection techniques (Amamra, et al., 2012)



	Advantages	Disadvantages
Static signature-based	Efficient and reliable to identify known malwares. Easy and simple to implement. Inexpensive computational calculation	Cannot detect unknown malwares and variants of known malwares. Easy evasion by using different obfuscation techniques
Dynamic behaviour signature-based	Detect entire family of malware by one signature. Dynamic signature is more close to malware behaviour.	Cannot detect new malware of a new behaviour. Dynamic behaviour signature must be accurate and compact.
Static behaviour signature-based	Detect entire family of malware with one signature. Detect malware before its execution.	Cannot detect unknown malwares. Computationally expensive.

CHAPTER III

3 RESEARCH METHODOLOGY

In this chapter, the research methodology is explained in detailed. The research methodology is designed and constructed by using the information obtained in the literature review phase. The analysis method used, feature selection, classification and clustering algorithms and the reasons why using those methods and algorithms are given as an informative part. The collection of data, the processes made on data, the implementation issues of algorithms and the evaluation of the results of algorithms are presented in the following sections.

3.1 Malware Detection Method

In the literature, there are two way of analysis to investigate the malware applications in mobile applications. Those analyses are static and dynamic analysis. The main differences of those detection methods are rely on the data used in the analysis. The data used in static analysis is extracted from the application code and other information about application which are not the runtime information. That is static analysis works without installing or running the application for the detection of malware. However, static analysis cannot detect the zero-day attacks that are an attack for which there is no corresponding signature. (Sudheer, et al., 2013) However, dynamic analysis requires run time information of application by running on the mobile device in a specific environment. The data is obtained after installing and running the mobile device by monitoring the activities realized by application on the mobile device. It is costly and time consuming and needs complicated skills. (Zhu & Peiravian, 2013)

The detection method selected in this study is the static analysis because the application is not executed; only analyzed in the static analysis. It has advantageous on limited memory of mobile devices. (Amos, et al., 2013) Another advantage of static analysis is undetectable process of analysis. Malware cannot detect the analyzer and cannot modify its behavior during analysis. (Apvrille & Strazzere, 2012)

3.2 Classification Algorithms and Feature Selection Methods

The classification algorithms and feature selection methods are explained briefly in this section. The advantages and disadvantages of those algorithms are discussed and why they are selected for the analysis to answer the research questions is expressed.

3.2.1 Classification Algorithms

Machine learning techniques have been widely applied for the detection of malicious applications in the literature (Schultz, et al., 2001), (Devesa, et al., 2010) and (Santos, et al., 2011). The studies about the detection of malware in the mobile platforms are mostly made by using the classification and clustering analysis. Yerima et al. (2013) utilized bayesian classification analysis in their android malware detection approach. The classifiers' performances were measured and evaluated in the detection of android malware in Amos et al. (2013). Zaw & Aung (2013) used both clustering and classification analysis in their permission based android malware detection study. Three decision tree algorithms; namely classification and regression tree (CART), J48 decision tree and random forest are selected in this study to answer research questions because decision trees are easy to understand and can classify both categorical and numerical data. In addition, there are no assumptions about the nature of data in decision tree algorithms. (Zhao & Zhang, 2007) Decision trees are non-parametric. The outliers and whether the data is linearly separable or not are not problem. The disadvantage of decision tree is that easily over fitting may occur. Random forest algorithm is an ensemble method overcoming this problem mostly. (Chen, 2011) In addition, Bayesian which is a probabilistic approach and SMO which is support vector classifier classification algorithms are selected to compare three types of classification algorithms in the detection malicious applications.

3.2.1.1 Bayesian Classification

This algorithm is based on Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data (George & Langley, 1995). Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability of belonging a given sample to a particular class. The naïve bayesian classifier is comparable with decision trees. Moreover, bayesian classifiers have exhibited high accuracy and speed when applied to large datasets. It assume that the effect of an attribute value on a class is independent of the values of other attributes (Han & Kamber, 2000) Bayesian classification is well suited to the problem of filtering large amounts of applications as it can perform relatively fast classification with low computational overhead once trained. The motivation in its implementation for detecting suspicious android applications is its ability of modeling expert and learning system easier than other machine learning techniques. (Yerima, et al., 2013) The figure 3.1 displays the details of Bayesian classifiers.

In statistical classification the Bayes classifier minimizes the probability of misclassification.

Suppose a pair (X,Y) takes values in $\mathbb{R}^d \times \{1,2,\ldots,K\}$, where Y is the class label of X. This means that the conditional distribution of X, given that the label Y takes the value r is given by

$$X \mid Y = r \sim P_r \text{ for } r = 1,2,\ldots,K$$

where "$\sim$" means "is distributed as", and where P_r denotes a probability distribution.

A classifier is a rule that assigns to an observation X=x a guess or estimate of what the unobserved label Y=r actually was. In theoretical terms, a classifier is a measurable function$C : \mathbb{R}^d \to \{1,2,\ldots,K\}$, with the interpretation that C classifies the point x to the class C(x). The probability of misclassification, or risk, of a classifier C is defined as

$$\mathcal{R}(C) = \mathrm{P}\{C(X) \neq Y\}.$$

The Bayes classifier is

$$C^{\mathrm{Bayes}}(x) = \underset{r \in \{1,2,\ldots,K\}}{\operatorname{argmax}} \ \mathrm{P}(Y = r \mid X = x).$$

Figure 3.1: Bayesian classifier (Wikipedia, 2013)

3.2.1.2 Classification and Regression Tree (CART)

This algorithm implements minimal cost-complexity pruning. It is a non-parametric decision tree learning technique that builds either classification or regression trees depending on the variable type; categorical or numeric respectively. Rule is selected to best splitting based on the variable values. After selecting the rule, a node is splitted into two nodes. This process is applied to each child node recursively. The splitting stops when CART detects no further gain. (Breiman, et al., 1998). The advantage of CART comes from the simplicity of results and the usage of three nonparametric and nonlinear methods. There is no implicit assumption that the relationship between predictor variable and dependent variable are linear. [Classification and Regression Trees (C$RT), 2014] CART discovers the interactions among variables. It is invariant of monotonic transformations of predictive variable and not sensitive to outliers in predictive variables. (Guszcza, 2005) In addition, the pruning method used by the CART for learning decision trees can often produce smaller trees than C4.5’s pruning method. (Witten, et al., 2011) The splitting and stopping rules for classification and regression tree algorithm is given in figure 3.2.

Splitting rule: Choose the split that maximizes the decrease in impurity. Impurity:

1. Gini Index

$$\phi(p) = \sum_{k \neq l} p_k p_l = 1 - \sum_{k \neq l} p_k^2$$

2. Entropy

$$\phi(p) = -\sum_{k} p_k \log p_k$$

Split stopping rule: A large tree is grown and procedures are implemented to prune the tree up-ward.

Class assignment: Normally simply assign the majority class in the node unless a strong prior of the class probability is available.

Figure 3.2: Classification and regression tree (CART) (Zaw & Aung, 2013)

3.2.1.3 J48 Decision Tree Algorithm

This algorithm generates pruned or unpruned C4.5 decision tree. C4.5 is an extension of Quinlan's earlier ID3 algorithm. C4.5 constructs decision trees from a set of training data by using the concept of information entropy. At each node, C4.5 selects the attribute of the data that most effectively splits its set of samples into subsets. Information gain is the splitting criterion used for splitting at each node while building the tree. The attribute with the highest gain is chosen to make the decision. J48 can be applied to dataset having both numerical and categorical attributes. (Quinlan, 1993) The details of J48 decision tree algorithm is shown in figure 3.3.

3.2.1.4 Random Forests (RF)

The idea behind random forest algorithm is to construct a forest of random trees. It is an ensemble learning method for classification. It works by building a decision tree at training time and outputting the class by individual trees. The training algorithm for random forests uses the general technique of bootstrap aggregating, or bagging, to the learners. The advantages of random forest algorithm are its efficiency on large databases, successfully handling thousands of input variables without deleting any variable. Moreover, it has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing. (Breiman, 2001) Random forest algorithm is selected in the analysis of permission based malware detection as a third decision tree algorithm. The steps of random forests algorithm is given in figure 3.4.

```
Input: training sample set T, the collection of candidate attribute attribute_list
Output: a decision tree.

1. Create a root node N;

2. If T belongs to the same category C, then return N as a leaf node, and mark it as class C;

3. If attribute_list is empty or the remainder samples of T is less than a given value, then return N as a leaf node, and mark it as the category which appears most frequently in attribute_list, for each attribute, calculate its information gain ratio

4. Suppose test_attribute is the testing attribute of N, then test_attribute= the attribute which has the highest information gain ratio in attribute list:

5. If testing attribute is continuous, then find its division threshold;

6. For each new leaf node grown by node N

        {
        a. Suppose T is the sample subset corresponding to the leaf node.

        b. If T has only a decision category, then mark the leaf node as this category;

        c. Else continue to implement J48_Tree (T',T'_attributelist)

        }
7. Calculate the classification error rate of each node and then prune the tree.
```

Figure 3.3: J48 decision tree algorithm (Zaw & Aung, 2013)

```
1. Select ntree, the number of trees to grow, and mtry, a number no larger than number of variables.

2. For i = 1 to ntree:

3. Draw a bootstrap sample from the data. Call those not in the bootstrap sample the "out-of-bag" data.

4. Grow a "random" tree, where at each node, the best split is chosen among mtry randomly selected variables. The tree is grown to maximum size and not pruned back.

5. Use the tree to predict out-of-bag data.

6. In the end, use the predictions on out-of-bag data to form majority votes.

7. Prediction of test data is done by majority votes from predictions from the ensemble of trees.
```

Figure 3.4: Random forest (RF) algorithm (Zaw & Aung, 2013)

3.2.1.5 Sequential Minimal Optimization (SMO)

Sequential minimal optimization algorithm presented by John Platt implements training of a support vector classifier. Training a support vector machine needs the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of smallest possible QP problems. This helps to avoid time consuming and to provide linear memory requirement with the training dataset size. This allows handling very large training sets. This algorithm is conceptually simple, easy to implement and is faster and better scaling properties than other SVM training algorithms. (Platt, 1998) SMO uses a set of heuristics. These heuristics are generally producing same or close decisions in practice. SMO treats linear SVMs in a special way giving a great speed up for training linear separators. (Joachims, 1998) The general algorithm processes are described as following;

$$f(x) = max_{\propto} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \propto_i \propto_j$$

Subject to:

$0 \leq \propto_i \quad \leq$ C for i = 1, 2, 3,…, n,

$\sum_{i=1}^{n} y_i \propto_j = 0$

Where C is an SVM hyperparameter and K(x_i, y_j) is a kernel function. The variables α_i are Langrange multipliers.

The algorithm proceeds;

1. Find a Langrange multiplier α_1 that violates the Karush-Kuhn-Tucker (KKT) conditions for the optimization problem.
2. Pick a second multiplier α_2 and optimize the pair (α_1, α_2)
3. Repeat steps 1 and 2 until convergence.

When all the Langrange multipliers satisfy the KKT conditions, the problem has been solved. The implementation of SMO with Weka globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default.

3.2.2 Feature Selection Methods

Feature selection methods are beneficial for the performance of classification algorithms because a large number of extracted features, some of which redundant or irrelevant, present several problems such as misleading the learning algorithm, over-fitting, reducing generality, and increasing model complexity and run-time. These bad effects are even more crucial when applying machine learning methods on mobile devices, since they are often restricted by processing and storage-capabilities. Applying fine feature selection before machine learning enabled to use malware detector more efficiently, with a faster detection. (Zaw & Aung, 2013)

Feature selection methods can be divided into two groups as attribute based and subset based feature selection methods. Attribute based feature selection methods evaluate each feature independent from other features. These methods need to a class feature to evaluate each feature with this class feature defined by the analyzer in attribute feature selection methods. The dependency of features to other features is out of consideration but the relation with class feature is under consideration in those feature selection methods. Gain Ratio and ReliefF attribute feature selection methods are this type of methods. In subset based feature selection methods, the subsets of features are constructed and the subset including some number of features which is best representing the whole features is selected by feature selection method. In these feature selection methods, the dependency of features to each other and class feature is considered in feature selection. Cfs and Consistency subset feature selection methods are used for subset type feature selection methods.

3.2.2.1 Gain Ratio Attribute Feature Selection Method

This method evaluates the worth of an attribute by measuring the gain ratio with respect to the class. Gain ratio attribute feature selection method requires a class feature to evaluate features. The benign/malware status is used for the class feature in this method. The attributes are evaluated with class attribute independently from other attributes. This method neglects the relation of attributes with each other. Also, the number of features that is desired to select is needed for this selection method. 25 and 50 features are determined as sufficient for the representation of 182 features. Hence, the number of features is separately reduced to 25 and 50 features by applying gain ratio feature selection method. The formula of Gain Ratio is given below.

GainR (Class, Attribute) = (H (Class) - H (Class | Attribute)) / H (Attribute).

3.2.2.2 ReliefF Attribute Feature Selection Method

This method evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class. It can operate on both discrete and continuous class data. (Kononenko, 1994) ReliefF attribute feature selection method is also requiring a class feature and a predefined feature number. The attributes are evaluated with class attribute independently from other attributes. This method neglects the relation of attributes with each other. Benign/Malware status is used as a class feature. 25 and 50 features are assumed to be satisfactory feature numbers for representing whole features.

3.2.2.3 Cfs Subset Feature Selection Method

This method evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred. (Hall, 1998) Cfs Subset feature selection method does not require a class feature because it selects subset of all features by using predefined feature number. The advantage of this method comes from the consideration of relation or dependency of attributes with each other. 25 and 50 feature numbers are used again in this feature selection method.

3.2.2.4 Consistency Subset Feature Selection Method

This method evaluates the worth of a subset of attributes by the level of consistency in the class values when the training instances are projected onto the subset of attributes. Consistency of any subset can never be lower than that of the full set of attributes. (Liu & Setiono, 1996) Consistency subset feature selection method does not need a predefined feature number. The method determines optimum number of feature and returns the optimum feature subset in the result. The advantage of this method comes from the consideration of relation or dependency of attributes with each other.

3.2.3 Clustering Algorithm

K-Means clustering algorithm is performed for clustering analysis in this study. The main idea here is to define k centroids, one for each cluster. The next step is to assign each application to the nearest centroid. K-means clustering partitions the dataset by minimizing the sum of squares cost function. (Abu Samra, et al., 2013) K-means clustering algorithm is data driven method relatively few assumptions on the distribution of underlying data. In addition, it guarantees at least local minimum of criterion function reducing the time for convergence of clusters on large datasets. (Zaw & Aung, 2013) K-means is a simple and easy way of clustering dataset into k clusters. (Abu Samra, et al., 2013) The details of K-means clustering algorithm is given in figure 3.5.

1. Select k random instances from the training data subset as the centroids of the clusters C1; C2; ...Ck.

2. For each training instance X:
    a. Compute the Euclidean distance D(Ci,X), i=1... k:Find cluster Cq that is closest to X.
    b. Assign X to Cq. Update the centroid of Cq.(The centroid of a cluster is the arithmetic mean of the instances in the cluster.)

3. Repeat Step 2 until the centroids of clusters C1;C2; ...Ck stabilize in terms of mean-squared- error criterion.

4. For each test instance Z:
    a. Compute the Euclidean distance D(Ci,Z), i=1... k. Find cluster Cr that is closest to Z.
    b. Classify Z as an anomaly or a normal instance using the Threshold rule

The Threshold rule for classifying a test instance Z that belongs to cluster Cr is: Assign Z-->1 if P(w|Z)> Threshold; Otherwise Z □0 where ―0‖ and ―1‖ represent normal and malware classes.

Figure 3.5: K-Means clustering algorithm (Zaw & Aung, 2013)

3.3 Data Collection

3.3.1 Apk Files

Mobile devices having android operating system are working with special applications which can be run only on the android mobile devices. The installation packets of those applications are named as apk files. Apk stands for Android Package Kit. Apk files are mainly available on the Google Play which consists of several types of android applications. The number of android applications is increasing very fast in recent years (Statista, 2014) and the number of downloads are measured with millions.

The data which is used in this study is obtained from COMODO Group, Inc. (COMODO Group, Inc., 2014) a branch of which is located at the Middle East Technical University campus in Turkey. They are working on the security issues on the mobile platforms. The dataset includes 3,784 apk files. 2,338 of them are benign apk files and 1,446 of them are malicious apk files. The label as ‘benign’ or ‘malware’ was provided in the dataset specified by the company. COMODO laboratory examines the application code of android applications to determine whether the android application is malware or not. The reason of why an application is malware is kept in their databases. For this reason, the status of apk files gathered from COMODO laboratory is reliable for the use of analysis. For this reason, benign or malware status about the android applications is accepted as it is given by COMODO. And this information is used in the analysis of the study.

Android applications are developed in Java programming language. The installation package of android applications is compressed (ZIP) bundle of files including AndroidManifest.xml (manifest file) and classes.dex files. The components of an android application such as the activities, services, broadcast receivers, and content providers are described in the manifest file. It declares which permissions the application must have in order to access protected parts of the API and interact with the applications’ components and other applications. (Developers, 2014) The system which is installing an android application must read and know that which components exist in the application and which permissions are required by looking at the manifest file. Those permissions defined in the manifest file should be accepted by the user before the android application is installed on the mobile device such as internet access, read sms, read mms and calling etc.

3.3.2 Permissions

Android applications are using some sources of mobile devices such as internet, sms, mms and calling etc. In order to use required sources, related permission about using that source must be given by the mobile device, actually by the user of mobile device while installing the android application. For this reason, mobile device is asked for the permissions required by that application before installing the application on mobile device. If the user accepts the permissions asked by the application; the application is then installed on the mobile device. The permissions required by the application determine the capability of installed application on the mobile device. Hence, those permissions required by the mobile device’s applications are valuable and worth to be investigated to detect malicious android applications. This type of investigation is classified as static analysis (Amamra, et al., 2012) because the analyzer does not install and run android program and only searches the effects of those permissions on the maliciousness of applications. The permissions required by the android applications are used in this paper as a data to answer the research questions whether the permissions have effect on the detection of malicious applications in android applications or not. The permission list given in Developers (2014) shows possible permissions in latest API level 20 that can be asked by the android applications. For example, “android.permission.CALL_PHONE” is one of those permissions and it allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed. Moreover, “android.permission.INTERNET” allows applications to open network sockets. The list of all permissions used in this study is given in appendix A.

3.3.3 Features

Features are the attributes used for defining the permission characteristics of android applications. Each feature corresponds to the one of the permission defined for that android application given in appendix A. All the features constitute feature vector which is defining all the permission properties of an android application. In addition to the permissions, version code and version name which are also data included in the manifest file is added to feature vector. The definition of feature vector is given in Figure 3.6. Each feature representing a permission is defined as binary number; 1 is for permission is required, 0 is for permission is not required. The exact data is used for version code and version name without any conversion. An example of feature vector for one apk file including 182 features (Version Code, Version Name and 180 Permissions) is given in Figure 3.7.

$$\boxed{f_i = \begin{cases} 1 \text{ if related permission is required} \\ \\ 0 \text{ if related permission is not required} \end{cases} \qquad i > 2}$$

Figure 3.6: Feature definition for the permissions

(3,1.2,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,0,1,1,
1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,0,0,0,1,1,0,0,0,1,0,1,0,1,1,0,0,0,0,0,0,1,1,0
,1,0,0,0,0,1,0,0,0,0,1,1,0,1,0,1,1,1,0,0,0,1,0,0,0,0,1,1,1,1,0,1,1,0,0,1,0,1,0,1,0,1,1,0,1,
0,1,1,1,1,1,1,0,0,1,0,0,1,1,1,1,1,0,0,1,1,0,0,1,0,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0
,0)

Figure 3.7: Feature vector example

3.3.4 Feature Extraction

Feature extraction is required for obtaining the permission data of android applications. Acquiring apk files is not adequate to use them in the analysis. apk files must be extracted and the AndroidManifest.xml file must be read to acquire permission data of apk files. For this reason, a feature extraction step should be performed before analysis while preparing the dataset. There are several feature extraction tools that can be seen in the literature. Some of those feature extraction tools are APK Tool (Open Source), dex2jar which is made by a Chinese student, Smali and dexdump dissembler. In this study, APK Tool is used for the extraction of apk files.

In order to reach the permission data of an android application, first the apk file must be extracted from .apk extended file as shown in figure 3.8 and then AndroidManifest.xml file must be read. In Google (2014), the APKTOOL is advised to extract apk file to android manifest file and smali files. After extracting the manifest file and smali files, the permissions defined in the android manifest file are examined to list which type of permissions are required to run the android application. This permission data is used in the analysis of detecting malware android applications. The figure that showing the feature extraction processes is given in below.

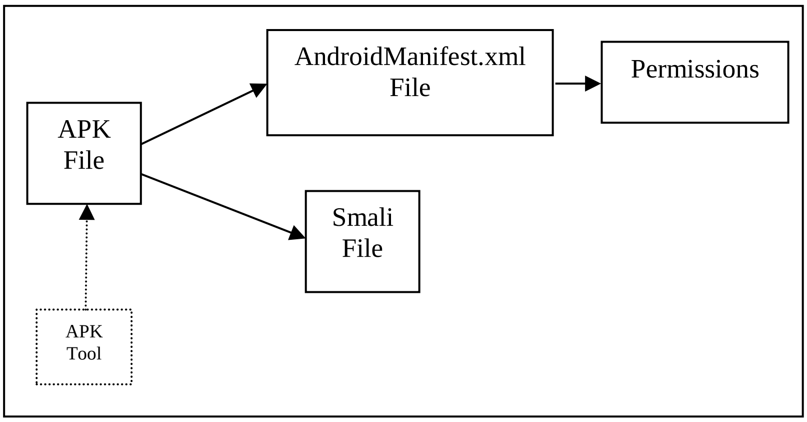


Figure 3.8: Feature extraction

3.4 Data Preprocess and Features for Machine Learning Algorithms

Weka (Waikato Environment for Knowledge Analysis) machine tool is made use of in the analysis of this study. Weka has the ability of preprocessing, classification, clustering, association, attribute selection and visualization. Some of those capabilities of weka are used for this study. The dataset in this analysis is a data matrix consisting of android applications in the rows and features in the columns. The values in each cell of this data matrix except version name and version code columns are binary number; 0 or 1. For this reason, no preprocessing step is needed in this study. Final dataset includes following features;

- Version code
- Version name
- All the permission listed in appendix A

3.5 Analysis of Feature Selection Methods and Classification Algorithms

In this part of analysis, classification algorithms given in section 3.2.1 and the feature selection methods given in section 3.2.2 are used to answer the following research questions;

- Which classification algorithm show better performance with higher accuracy for the detection of malware applications among Bayesian, Classification and Regression Tree (CART), J48 Decision Tree, Random Forest and Sequential Minimal Optimization (SMO)?

- Which classification algorithm, feature selection method and the number of selected features combination is more accurate in detection of malware in android applications?

In order to answer those research questions, the steps given in figure 3.9 are implemented. The analysis is performed by using dataset having all 3,784 android applications. First of all, feature selection methods are applied separately to the dataset. After selecting most appropriate features by using 4 feature selection methods, 7 datasets, consisting 25 and 50 features for 3 feature selection methods namely; gain ratio attribute, reliefF attribute and cfs subset, and optimum number of features for consistency subset feature selection method, are formed which consist of all android applications and selected features. Then, 5 classification algorithms are applied to those prepared datasets. Hence, totally 35 iterations are performed with defined steps in the following figure.

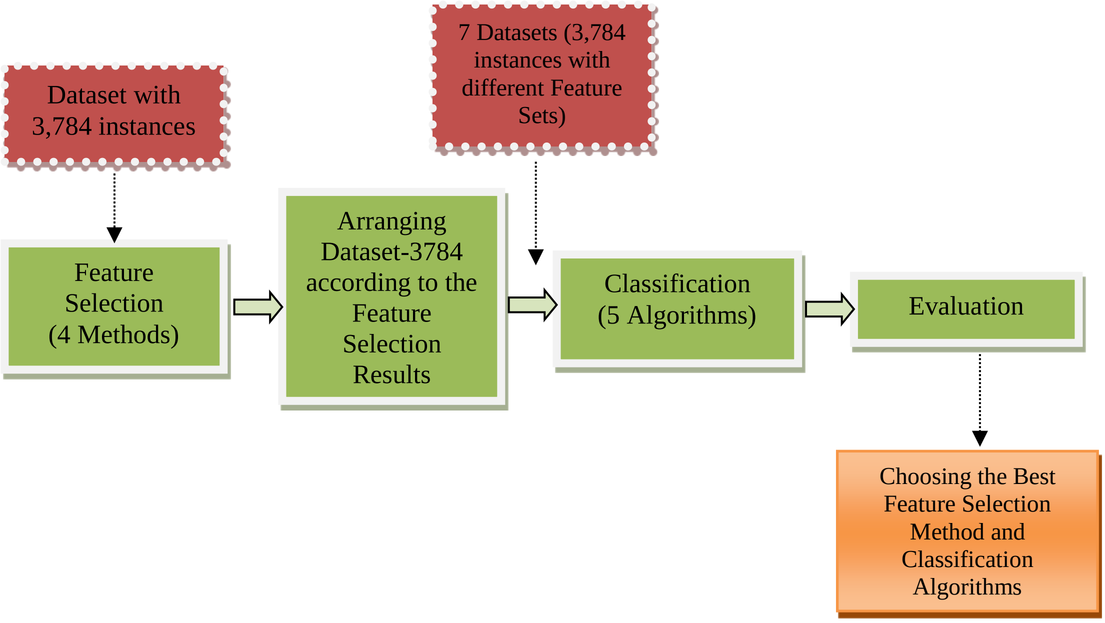

Dataset with 3,784 instances
7 Datasets (3,784 instances with different Feature Sets)
Feature Selection (4 Methods)
Arranging Dataset-3784 according to the Feature Selection Results
Classification (5 Algorithms)
Evaluation
Choosing the Best Feature Selection Method and Classification Algorithms

Figure 3.9: The details of feature selection method and classification algorithms implementation

3.6 Analysis of Clustering Algorithm

K-means clustering algorithm is used in the analysis of clustering in order to answer third research question; is clustering algorithm useful on determination of which permissions have more tendencies to be used by malicious applications? By using k-means clustering algorithm, the dataset including 3,784 instances is divided into 3 clusters. The characteristics of each cluster are investigated according to the results of clustering implementation. The number of benign and malware applications are examined to determine the tendencies of clusters. The features are analyzed in each cluster whether they are mostly used by malicious applications or not. That is which features are mostly demanded by malicious applications are determined in order to answer research question. The steps of analysis implemented for clustering analysis are shown in Figure-3.10.

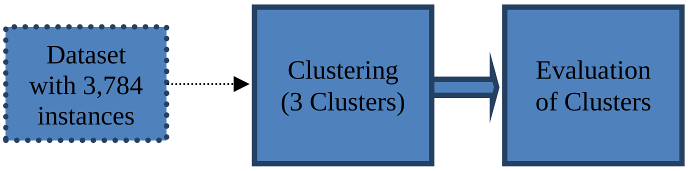


Figure 3.10: General view of clustering analysis and evaluation of clusters

3.7 Analysis of Dataset Size

Datasets having different number of instances are constructed for the analysis of classification algorithms. By this way, it is intended to investigate the last research question; is the dataset size effective on the performance of classification algorithms? The number of instances for each dataset is chosen as 500, 1,000, 2,500 and 3,784 instances. The benign/malware status of apk files are known in the dataset-3784 (representing the dataset having 3,784 instances). The number of benign and malware apk files in the datasets 500, 1,000 and 2,500 are selected equivalent to each other. That is in the dateaset-500, there are 250 benign APK files and 250 malware APK files. The same thing is also viable for the dataset-1000 and dataset-2500. However, dataset-3784 does not have equal number of benign and malware apk files, it includes 2,338 benign apk files and 1,446 malware apk files. In addition, the datasets cover each other because we add new instances to previously formed dataset. For instance, dataset-1000 is formed by adding randomly selected 250 benign apk files and 250 malware apk files to dataset-500. Dataset-1000 covers the dataset-500, dataset-2500 covers the dataset-1000 and dataset-3784 covers the dataset-2500. Prepared datasets are used to answer the research question about the effect of dataset size on malware detection.

Feature selection method is applied to prepared datasets for the selection of most appropriate features for the analysis of classification. According to the results of feature selection method implementation, the datasets are rearranged by getting rid of non-selected features. The apk files are kept as it is. Only some features that are considered as irrelevant or misleading are removed from the 4 datasets. By this way, datasets on hand are differentiated by their feature properties as well as their apk file contents.

After feature selection step, the datasets are ready for the implementation of classification algorithms. The classification algorithms are applied to datasets having different number of instances and different feature properties. Hence, we will be able to evaluate the performance of classification algorithms on different dataset sizes. The results of classification analysis are evaluated to find answer to last research question. General view of analysis made in this part of analysis is given in Figure-3.11.

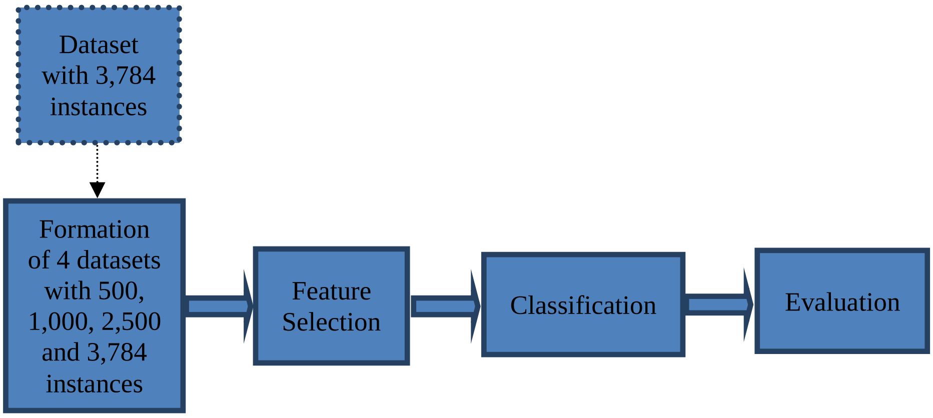

Dataset with 3,784 instances
Formation of 4 datasets with 500, 1,000, 2,500 and 3,784 instances
Feature Selection
Classification
Evaluation

Figure 3.11: General view of analysis for the effect of dataset size on the classification algorithms

3.8 Evaluation Measures for Classification Algorithms

The performance evaluation of feature selection methods and classification algorithms are made in this part of analysis to determine the best performing feature selection method and classification algorithm. The evaluation of classification algorithm implementations are made by looking at Overall Accuracy (ACC), TP Rate (TPR), FP Rate (FPR), Precision (PPV), Recall and F-Measure values in the literature which are the output of classification algorithms. (Yerima, et al., 2013), (Zaw & Aung, 2013)

True Positive (TP): Number of correctly identified benign applications.
False Positive (FP): Number of wrongly identified malware applications.
True Negative (TN): Number of correctly identified malware applications.
False Negative (FN): Number of wrongly identified benign applications.

Table 3.1 below summarizes the four basic classification measures described above.

Table 3.1: Confusion Matrix

		Prediction	
		Malware	**Benign**
Reality	**Malware**	TRUE NEGATIVE	FALSE POSITIVE
	Benign	FALSE NEGATIVE	TRUE POSITIVE

True Positive Rate (TPR): Percentage of correctly identified benign applications. True Positive Rate is also named as Recall.
TPR = TP / (TP+FN)

False Positive Rate (FPR): Percentage of wrongly identified malware applications.
FPR = FP / (TN+FP)

Overall Accuracy (ACC): Percentage of correctly identified applications. In other words, it is the correctly classified instances.
ACC = (TP+TN) / (TP+TN+FP+FN)

Precision (PPV): (also called positive predictive value) is the fraction of retrieved instances that are relevant.
PPV = TP / (TP + FP)

F-Measure: A measure that combines precision and recall is the harmonic mean of precision and recall.

In this study, only overall accuracy and precision is considered in the evaluation of analysis results.

CHAPTER IV

4 RESULTS AND EVALUATION

In this chapter, the evaluation of the results for the classification algorithms, clustering algorithms and feature selection methods is presented. The answers for the research questions are tried to be found in this chapter. Firstly, the classification algorithms giving the most accurate results are discussed. Then, feature selection methods and the compatibility with classification algorithms are evaluated. In addition, the features are examined to determine the tendency of them used by malicious applications with clustering analysis. Lastly, the effect of dataset sizes is evaluated according to the results of classification implementations.

4.1 Brief Information about Datasets and Configuration of Algorithms

Dataset having 3,784 instances is used for the analysis made in this study. Dataset is a data matrix including android applications in the rows and features in the columns. Dataset is rearranged in the analysis by changing the number of android applications in the rows and the features in the columns in order to test the issues defined in research questions. 25 and 50 features used in the analysis except consistency subset feature selection method because this method determines the optimum feature number in its implementation. The higher number of features in the dataset increases the complexity of analysis and time spent on it. Some of the 182 features in the dataset may be redundant or unnecessary and may mislead the classification algorithm results. More than one feature number is used in feature selection step because 25 features may not be enough to represent whole features or 50 features may include redundant features or misleading features. The feature content and feature number is arranged according to the results of feature selection method implementations. The number of android application is changed in order to analyze the effect of dataset size on the classification algorithms for the detection of malware.

The analysis of feature selection methods and classification algorithms are performed with 35 iterations [1 dataset x 7 feature selection methods with feature numbers (1 feature number for consistency subset) x 5 classification algorithms]. 25 and 50 features are used for three feature selection methods. Consistency subset feature selection method determines feature number within the method implementation. Weka machine learning tool is utilized in applying classification algorithms. The algorithms in weka presents default option settings in the implementation of classification algorithms. Those default options may be changed by the user if it is needed. But in this study, the default options presented by weka machine learning tool for feature selection method and classification algorithms are kept as it is.

In order to limit the problems like over fitting and variability in the results, k-fold cross validation is used to divide training and testing data in classification of datasets. According to the Keller, k-fold cross validation maximizes the use of the data.

- Divide data randomly into k folds (subsets) of equal size.
- Train the model on k-1 folds, use one fold for testing.
- Repeat this process k times so that all folds are used for testing.
- Compute the average performance on the k test sets.

This effectively uses all the data for both training and testing. In the present study, we assumed k = 5 for the k value.

In clustering analysis, k-means clustering algorithm used with k = 3. The distance function used for clustering is euclidean distance. The analysis made on the training set only. By using the clustering results, the evaluation of features is realized by measuring the tendency of their existence in malware instances. The default options presented by the weka machine learning tool k-means clustering algorithm are kept as it is.

Dataset size effect is measured by preparing the 4 datasets from the dataset on hand. Firstly, 500 instances are selected randomly with 250 benign instances and 250 malicious instances. Secondly, 250 benign and 250 malicious instances are added on the dataset having 500 instances in order to obtain dataset having 1,000 instances. Then, 750 benign and 750 malicious instances are added on the dataset having 1,000 instances to get dataset having 2,500 instances. Lastly, dataset having 3,784 instances is used as fourth dataset. Those datasets are used for measuring the effect of dataset size on the classification algorithms. 8 iterations (4 datasets x 1 feature selection method x 1 feature number x 2 classification algorithms) of classification algorithm implementation is realized in this part of analysis. The default options presented by the weka machine learning tool for feature selection method and classification algorithms are kept as it is. Moreover, 5 fold cross validation is used for the construction of training and testing sets.

4.2 Evaluation of Classification Algorithms

In this part of the study, the classification algorithm results are investigated to find answer to first research question. The overall accuracy results of classification algorithms among feature selection methods are compared for Bayesian algorithm, CART, J48 Decision Tree, Random Forest and SMO algorithms. The results are evaluated by using overall accuracies and paired t-test results. The results of classification algorithms are given in Appendix-B.

As it can be seen from the figure 4.1, all the overall accuracy values for Bayesian algorithm are below 90 %. The overall accuracy among all feature selection methods are close except consistency subset feature selection method in which it gives around 73 % overall accuracy. The values within the parenthesis in the name of feature selection methods represent the selected features. For example, Cfs (25) means that 25 features selected with Cfs subset feature selection method.

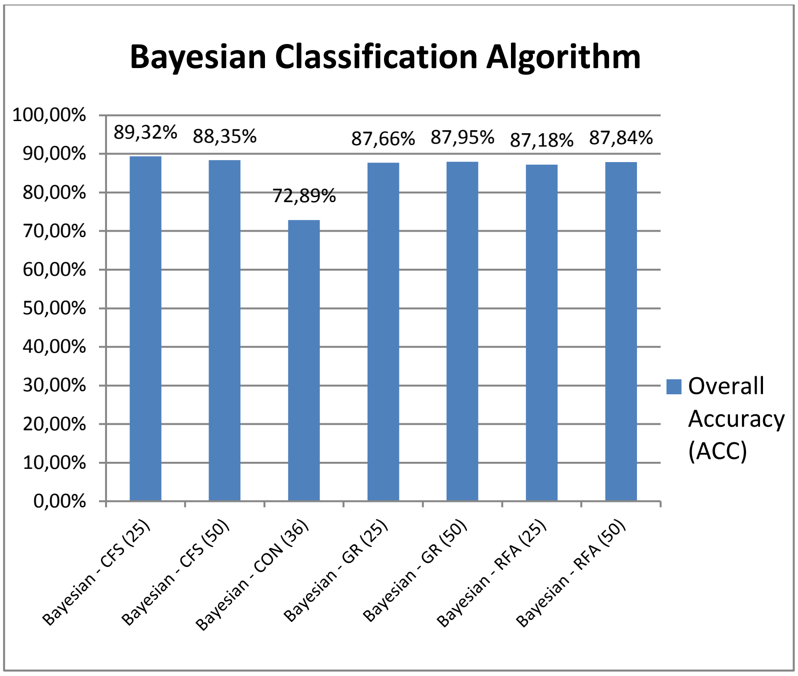


Figure 4.1: Overall accuracy values for Bayesian classification algorithm

The comparison of Bayesian algorithm is realized by applying paired t-test (Table 4.1) among datasets selected features with different feature selection methods. When the results of all t-test result are evaluated, it can be said that all other classification algorithm's accuracy results are better than the Bayesian accuracy with statistically 0.05 confidence level in all datasets implemented different feature selection methods. Hence, the worst performance among classification algorithms belongs to the Bayesian classification algorithm.

Table 4.1: Comparison of prediction accuracy of Bayesian algorithm with others

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with Cfs Subset (25)	89.24	93.65 v	94.34 v	93.55 v	91.83 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with Cfs Subset (50)	88.46	90.50 v	92.84 v	90.47 v	92.57 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with Consistency Subset (36)	72.79	73.38	74.04 v	73.88 v	72.69

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with Gain Ratio Attribute (25)	87.77	92.10 v	92.12 v	91.96 v	91.19 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with Gain Ratio Attribute (50)	87.89	94.06 v	94.55 v	93.80 v	92.74 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with ReliefF Attribute (25)	87.24	90.16 v	92.05 v	90.35 v	92.50 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Bayesian	J48	Random Forest	CART	SMO
Dataset-3784 with ReliefF Attribute (50)	87.84	91.43 v	93.55 v	90.40 v	93.34 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification and regression tree shows second best performance in one feature selection method. The prediction accuracies among different feature selection methods can be seen from figure 4.2. The accuracy values are 90 % or over it with all feature selection methods except consistency subset feature selection method. CART shows best performance with CFS (25) and Gain Ratio Attribute (50) feature selection methods around 93.60 % prediction accuracy.

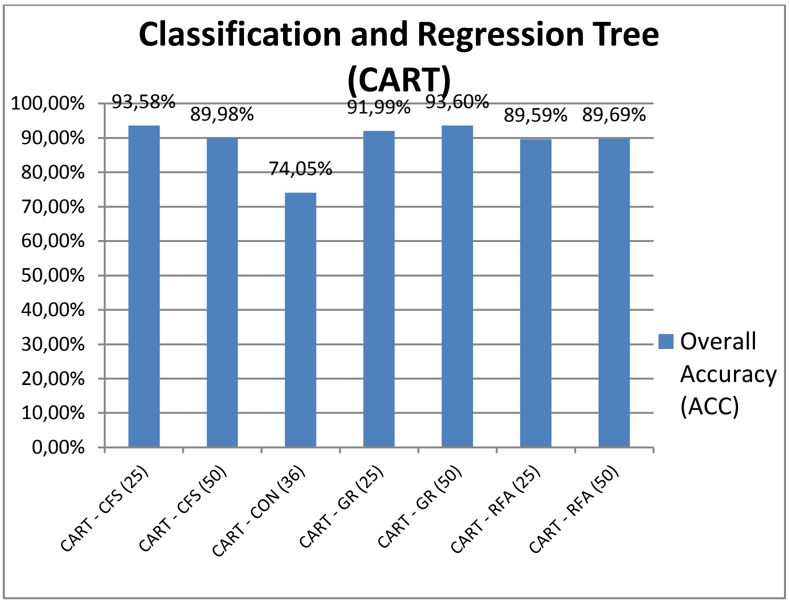


Figure 4.2: Overall accuracy values for Classification and Regression Tree (CART) algorithm

The t-test results (Table 4.2) show that CART is better than the Bayesian in all feature selection methods. The accuracy results of CART and J48 is not different statistically with 0.05 confidence level. Random forest algorithm is generally better than CART. Moreover, CART gives both statistically better and worse results than SMO in different feature selection methods.

Table 4.2: The comparison of the prediction accuracy of CART with others

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with Cfs Subset (25)	93.55	93.65	94.34 v	89.24 *	91.83 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with Cfs Subset (50)	90.47	90.50	92.84 v	88.46 *	92.57 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with Consistency Subset (36)	73.88	73.38	74.04	72.79 *	72.69 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with Gain Ratio Attribute (25)	91.96	92.10	92.12	87.77 *	91.19

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with Gain Ratio Attribute (50)	93.80	94.06	94.55 v	87.89 *	92.74 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with ReliefF Attribute (25)	90.35	90.16	92.05 v	87.24 *	92.50 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	CART	J48	Random Forest	Bayesian	SMO
Dataset-3784 with ReliefF Attribute (50)	90.40	91.43	93.55 v	87.84 *	93.34 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

The overall accuracy values with J48 decision tree algorithm are generally above the 90 % according to the figure 4.3. The best accuracy results are taken with Cfs subset (25) and gain ratio attribute (50) feature selection methods. When J48 decision algorithm is compared with other algorithms by t-test results (Table 4.3), random forest algorithm gives better results than J48 with 0.05 confidence level statistically. There is no statistically difference between J48 and CART algorithms. J48 shows better performance than Bayesian.

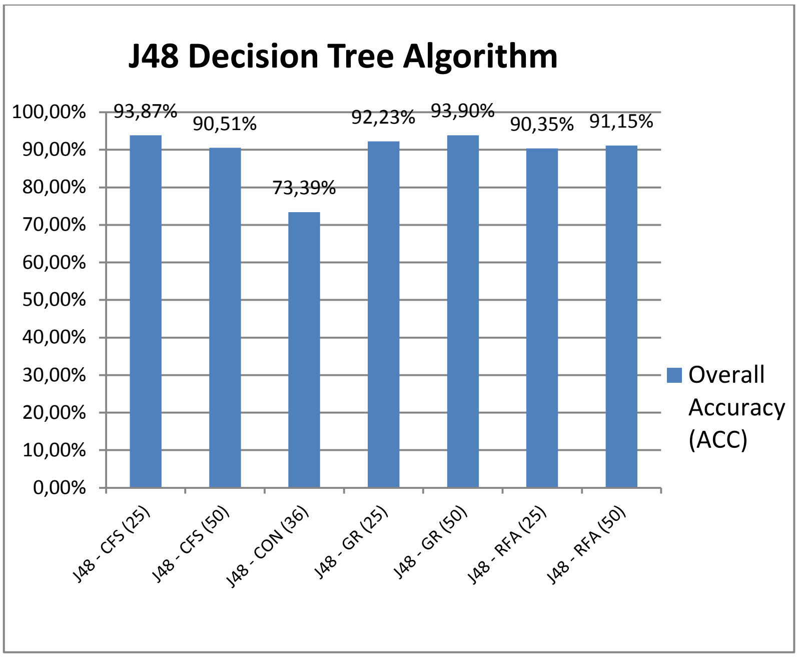


Figure 4.3: Overall accuracy values for J48 decision tree algorithm

Table 4.3: The comparison of the prediction accuracy of J48 with others

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with Cfs Subset (25)	93.65	94.34	93.55	89.24 *	91.83 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with Cfs Subset (50)	90.50	92.84 v	90.47	88.46 *	92.57 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with Consistency Subset (36)	73.38	74.04	73.88	72.79	72.69

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with Gain Ratio Attribute (25)	92.10	92.12	91.96	87.77 *	91.19 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with Gain Ratio Attribute (50)	94.06	94.55	93.80	87.89 *	92.74 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with ReliefF Attribute (25)	90.16	92.05 v	90.35	87.24 *	92.50 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	J48	Random Forest	CART	Bayesian	SMO
Dataset-3784 with ReliefF Attribute (50)	91.43	93.55 v	90.40	87.84 *	93.34 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

As it can be seen from figure 4.4, the overall accuracy values are mostly around 92 % and above it in random forest algorithm. It gives best results in five datasets.

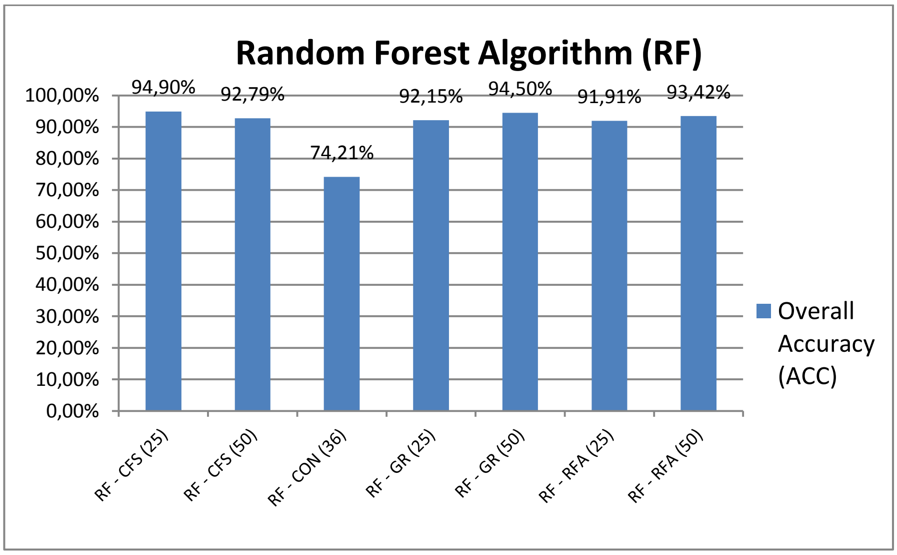


Figure 4.4: Overall accuracy values for Random Forest (RF) algorithm

T-test results show that random forest algorithm is statistically better than other algorithms in all feature selection methods. In some of the comparisons, it gives no difference with some of algorithms.

Table 4.4: The comparison of the prediction accuracy of Random Forest with others

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with Cfs Subset (25)	94.34	93.65	93.55 *	89.24 *	91.83 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with Cfs Subset (50)	92.84	90.50 *	90.47 *	88.46 *	92.57

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with Consistency Subset (36)	74.04	73.38	73.88	72.79 *	72.69 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with Gain Ratio Attribute (25)	92.12	92.10	91.96	87.77 *	91.19 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with Gain Ratio Attribute (50)	94.55	94.06	93.80 *	87.89 *	92.74 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with ReliefF Attribute (25)	92.05	90.16 *	90.35 *	87.24 *	92.50

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	Random Forest	J48	CART	Bayesian	SMO
Dataset-3784 with ReliefF Attribute (50)	93.55	91.43 *	90.40 *	87.84 *	93.34

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

The accuracy values are between 91 % and 93 % for SMO in most of the feature selection methods. Best results are obtained with reliefF attribute feature selection method.

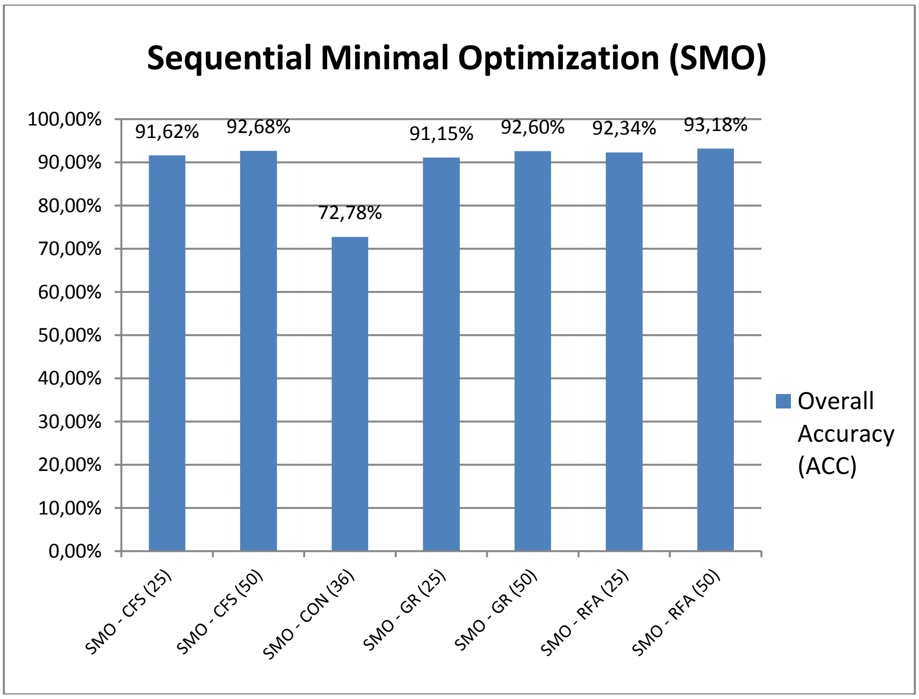


Figure 4.5: Overall accuracy values for Random Forest (RF) algorithm

The performance of SMO is statistically better than Bayesian and worse than random forest algorithm with 0.05 confidence level according to the t-test results. In reliefF attribute feature selection method, it gives statistically better results than other algorithms except Random Forest algorithm.

Table 4.5: The comparison of the prediction accuracy of SMO with others

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with Cfs Subset (25)	91.83	93.65 v	94.34 v	93.55 v	89.24 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with Cfs Subset (50)	92.57	90.50 *	92.84	90.47 *	88.46 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with Consistency Subset (36)	72.69	73.38	74.04 v	73.88 v	72.79

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with Gain Ratio Attribute (25)	91.19	92.10 v	92.12 v	91.96	87.77 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with Gain Ratio Attribute (50)	92.74	94.06 v	94.55 v	93.80 v	87.89 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with ReliefF Attribute (25)	92.50	90.16 *	92.05	90.35 *	87.24 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Dataset	SMO	J48	Random Forest	CART	Bayesian
Dataset-3784 with ReliefF Attribute (50)	93.34	91.43 *	93.55	90.40 *	87.84 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Table 4.6: Paired t-test ranking results among classification algorithms

Rank	Cfs Subset (25)	Cfs Subset (50)	Consistency Subset (36)	Gain Ratio Attribute (25)	Gain Ratio Attribute (50)	ReliifF Attribute (25)	ReliifF Attribute (50)
1.	Random Forest	SMO Random Forest	CART Random Forest	Random Forest J48	Random Forest	SMO Random Forest	SMO Random Forest
2.	J48	CART J48	J48	CART	J48	CART J48	CART J48
3.	CART	Bayesian	SMO Bayesian	SMO	CART	Bayesian	Bayesian
4.	SMO			Bayesian	SMO		
5.	Bayesian				Bayesian		

The answer of first research question is the random forest and J48 decision tree algorithms. According to the ranking results of t-test shown in table 4.6, random forest algorithm shows better performance than other algorithms without a shadow of a doubt. J48 decision tree algorithm is racing with the SMO but it can be seen from the table 4.6 that J48 decision tree beats SMO in four datasets. Also, J48 gives best accuracy result after random forest. Bayesian algorithm gives the worse results when compared with other algorithms according to ranking results.

4.3 Evaluation of Feature Selection Methods and Compatibility with Classification Algorithms

The performance of feature selection methods and the compatibility of them with classification algorithms are discussed in this part of the study in order to find answer to second research question. The performance of classification algorithms shows difference according to the used feature selection method. In the remaining part of this section, the performance of feature selection methods is discussed with the accuracy performance of classification algorithms. Paired t-test is used to statistically compare the classification results in this part as well.

The overall accuracy values for classification algorithms implemented to the datasets in which features are selected with Cfs subset feature selection method are shown in figure 4.6. As it can be seen from this figure, Random forest, J48 and CART algorithms show good performance with Cfs subset feature selection method. In addition, Cfs subset gives better performance in classification algorithms with 25 features except SMO implementation. This means that 25 features are enough and better representing the whole features. That is additional 25 features within 50 features is not beneficial for the performance of classification algorithms with Cfs subset feature selection method.

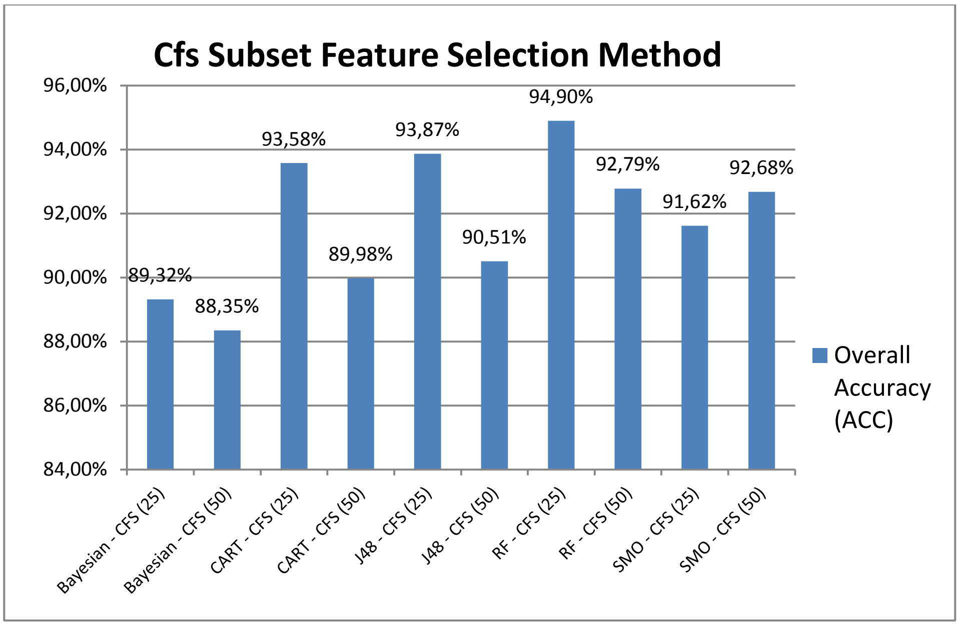


Figure 4.6: Overall accuracy values of classification algorithms with Cfs subset feature selection method

Paired t-test is applied to datasets in which feature selection method are implemented to reduce the feature sizes. According to the t-test results shown in table 4.7, Cfs (25) shows best performance in all classification algorithms except SMO because t-test gives that other feature selection methods statistically are worse than the Cfs (25).

Table 4.7: The comparison of prediction accuracy of classification algorithms by using Cfs (25) with other feature selection methods

Classification Alg.	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
Bayesian	89.32	88.35 *	72.89 *	87.66	87.95 *	87.18 *	87.84 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
CART	93.58	89.96 *	74.05 *	91.99 *	93.60	89.61 *	89.72 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
J48	93.87	90.51 *	73.39 *	92.23 *	93.90	90.35 *	91.15 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
Random Forest	94.29	92.84 *	74.23 *	91.99 *	94.82	92.13	93.21

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
SMO	91.62	92.68	72.78 *	91.15	92.60	92.34	93.18 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Cfs (50) gives the best result with SMO classification algorithm. In other classification algorithms, it is worse than Cfs (25). The paired t-test results are given in below table.

Table 4.8: The comparison of prediction accuracy of classification algorithms by using Cfs (50) with other feature selection methods

Classification Alg.	Cfs (50)	Cfs (25)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
Bayesian	88.35	89.32 v	72.89 *	87.66	87.95	87.18 *	87.84

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (50)	Cfs (25)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
CART	89.96	93.58 v	74.05 *	91.99 v	93.60 v	89.61	89.72

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (50)	Cfs (25)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
J48	90.51	93.87 v	73.39 *	92.23 v	93.90 v	90.35	91.15

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (50)	Cfs (25)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
Random Forest	92.84	94.29 v	74.23 *	91.99	94.82	92.13	93.21

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Cfs (50)	Cfs (25)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
SMO	92.68	91.62	72.78 *	91.15 *	92.60	92.34	93.18

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

The overall accuracy values for Gain ratio attribute feature selection method are shown in figure 4.7. In this figure, gain ratio attribute feature selection method with 25 and 50 features gives around 90 % and over it in all classification algorithms except Bayesian. Gain ratio attribute feature selection with 50 features shows better performance than 25 features in all classification algorithms. This means that additional 25 features is good for classification.

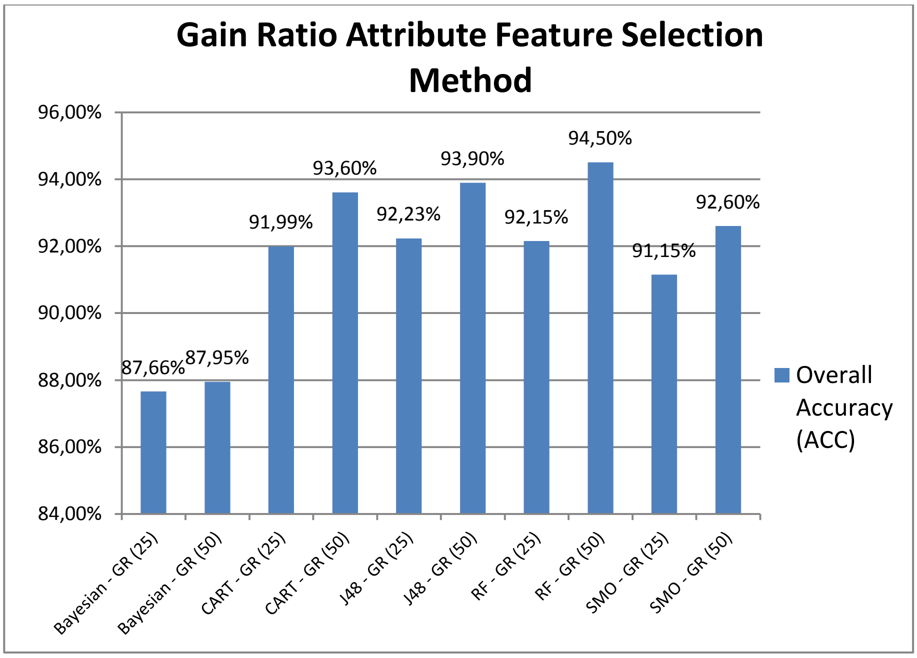


Figure 4.7: Overall accuracy values of classification algorithms with Gain Ratio Attribute feature selection method

Random forest, J48 and CART classification algorithms shows good performance with Gain ratio attribute feature selection method with 50 features according to the t-test results given in table 4.9. The classification algorithms shows statistically worse performance with Gain (25) than with Gain (50) and Cfs (25) according to t-test results with 0.05 confidence level.

Table 4.9: The comparison of prediction accuracy of classification algorithms by using Gain (25) with other feature selection methods

Classification Alg.	Gain (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (50)	ReliefF (25)	ReliefF (50)
Bayesian	87.66	89.32	88.35	72.89 *	87.95	87.18	87.84

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (50)	ReliefF (25)	ReliefF (50)
CART	91.99	93.58 v	89.96 *	74.05 *	93.60 v	89.61	89.72 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (50)	ReliefF (25)	ReliefF (50)
J48	92.23	93.87 v	90.51 *	73.39 *	93.90 v	90.35 *	91.15

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (50)	ReliefF (25)	ReliefF (50)
Random Forest	91.99	94.29 v	92.84	74.23 *	94.82 v	92.13	93.21

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (50)	ReliefF (25)	ReliefF (50)
SMO	91.15	91.62	92.68 v	72.78 *	92.60	92.34	93.18 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

The classification algorithms show statistically better performance with Gain (50) in all feature selection methods except Cfs (25). There is no statistically difference between Gain (50) and Cfs (25) feature selection methods on the performance of classification algorithms. The paired t-test results for Gain (50) feature selection method are given in below table.

Table 4.10: The comparison of prediction accuracy of classification algorithms by using Gain (50) with other feature selection methods

Classification Alg.	Gain (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	ReliefF (25)	ReliefF (50)
Bayesian	87.95	89.32 v	88.35	72.89 *	87.66	87.18	87.84

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	ReliefF (25)	ReliefF (50)
CART	93.60	93.58	89.96 *	74.05 *	91.99 *	89.61 *	89.72 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	ReliefF (25)	ReliefF (50)
J48	93.90	93.87	90.51 *	73.39 *	92.23 *	90.35 *	91.15 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	ReliefF (25)	ReliefF (50)
Random Forest	94.82	94.29	92.84	74.23 *	91.99 *	92.13 *	93.21 *

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Gain (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	ReliefF (25)	ReliefF (50)
SMO	92.60	91.62	92.68	72.78 *	91.15	92.34	93.18

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Random forest (RF) classification algorithm shows the best result with ReliefF attribute feature selection method with 50 features. Also, SMO classification algorithm is best compatible with the relief attribute feature selection method. 50 features is more beneficial than 25 features in reliefF attribute feature selection because classification algorithms give better performance with 50 features as shown in figure 4.8.

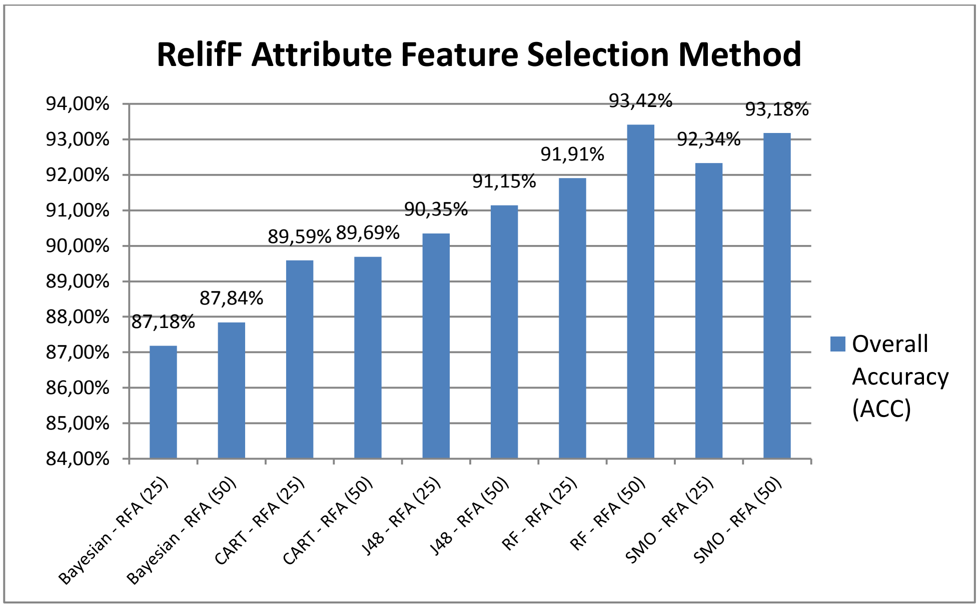


Figure 4.8: Overall accuracy values of classification algorithms with ReliefF Attribute feature selection method

There is no statistically difference with 0.05 confidence level between reliefF (25) and reliefF (50) on the performance of classification algorithms. The contribution of reliefF (25) on the performance of classification algorithms is not statistically better than other feature selection methods according to the t-test result given below table.

Table 4.11: The comparison of prediction accuracy of classification algorithms by using ReliefF (25) with other feature selection methods

Classification Alg.	ReliefF (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (50)
Bayesian	87.18	89.32 v	88.35 v	72.89 *	87.66	87.95	87.84

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (50)
CART	89.61	93.58 v	89.96	74.05 *	91.99	93.60 v	89.72

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (50)
J48	90.35	93.87 v	90.51	73.39 *	92.23 v	93.90 v	91.15

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (50)
Random Forest	92.13	94.29	92.84	74.23 *	91.99	94.82 v	93.21

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (25)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (50)
SMO	92.34	91.62	92.68	72.78 *	91.15	92.60	93.18

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

There is no statistically difference between reliefF (50) and reliefF (25), Cfs (50) feature selection methods for the performance of classification algorithms. Gain (50) and Cfs (25) gives better performance than reliefF (50) statistically with 0.05 confidence level. SMO shows good performance with both reliefF (25) and reliefF (50) feature selection methods.

Table 4.12: The comparison of prediction accuracy of classification algorithms by using ReliefF (50) with other feature selection methods

Classification Alg.	ReliefF (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)
Bayesian	87.84	89.32 v	88.35	72.89 *	87.66	87.95	87.18

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)
CART	89.72	93.58 v	89.96	74.05 *	91.99 v	93.60 v	89.61

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)
J48	91.15	93.87 v	90.51	73.39 *	92.23	93.90 v	90.35

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)
Random Forest	93.21	94.29	92.84	74.23 *	91.99	94.82 v	92.13

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	ReliefF (50)	Cfs (25)	Cfs (50)	Con (36)	Gain (25)	Gain (50)	ReliefF (25)
SMO	93.18	91.62 *	92.68	72.78 *	91.15 *	92.60	92.34

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

All the classification algorithms shows bad performance with consistency subset feature selection method as it can be seen from figure 4.9. The accuracy values for classification algorithms are lower than 75 %. The reason of these results might be the features selected by consistency subset feature selection method which are not much existed in the malicious applications. The classification algorithms do not give good results with those features. This does not mean consistency subset feature selection method is not good method for feature selection but it is not convenient to use with classification algorithms used in this study.

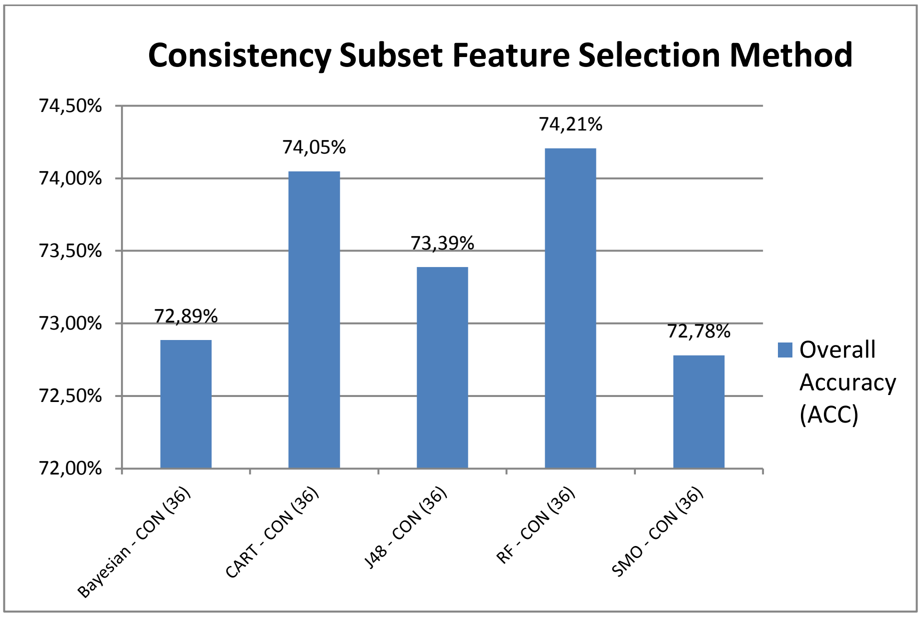


Figure 4.9: Overall accuracy values of classification algorithms with Consistency Subset feature selection method

The contribution of consistency subset feature selection method on the classification performance is statistically worse than other feature selection methods according to the t-test results given in table 4.13.

Table 4.13: The comparison of prediction accuracy of classification algorithms by using Consistency (36) with other feature selection methods

Classification Alg.	Con (36)	Cfs (25)	Cfs (50)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
Bayesian	72.89	89.32 v	88.35 v	87.66 v	87.95 v	87.18 v	87.84 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Con (36)	Cfs (25)	Cfs (50)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
CART	74.05	93.58 v	89.96 v	91.99 v	93.60 v	89.61 v	89.72 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Con (36)	Cfs (25)	Cfs (50)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
J48	73.39	93.87 v	90.51 v	92.23 v	93.90 v	90.35 v	91.15 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Con (36)	Cfs (25)	Cfs (50)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
Random Forest	74.23	94.29 v	92.84 v	91.99 v	94.82 v	92.13 v	93.21 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

Classification Alg.	Con (36)	Cfs (25)	Cfs (50)	Gain (25)	Gain (50)	ReliefF (25)	ReliefF (50)
SMO	72.78	91.62 v	92.68 v	91.15 v	92.60 v	92.34 v	93.18 v

v: statistically better than the compared value at 0.05 confidence level
*: statistically worse than the compared value at 0.05 confidence level

When the general results of the classification algorithms are evaluated, Cfs subset (25) feature selection method gives good performance in 3 classification algorithms which are Bayesian, J48 Decision Tree and Random Forest algorithms. Another important outcome of this analysis is that Cfs subset (25) shows better performance than Cfs subset (50). This means that 25 feature selected with Cfs subset feature selection method is enough and better representing whole features. In addition, it can be said that other 25 features selected in Cfs subset (50) feature selection method is redundant and misleading features.

Table 4.14: Paired t-test ranking results among feature selection methods

Rank	Bayesian	CART	J48	Random Forest	SMO
1.	Cfs (25)	Gain (50) Cfs (25)	Gain (50) Cfs (25)	Gain (50)	ReliefF (50)
2.	Gain (25)	Gain (25)	Gain (25)	Cfs (25)	Cfs (50)
3.	Cfs (50)	ReleifF (25)	ReliefF (50)	ReliefF (50) ReliefF (25) Cfs (50)	ReliefF (25) Gain (50)
4.	ReliefF (50) Gain (50)	ReliefF (50) Cfs (50)	ReliefF (25) Cfs (50)	Gain (25)	Cfs (25)
5.	ReliefF (25)	Consistency (36)	Consistency (36)	Consistency (36)	Gain (25)
6.	Consistency (36)				Consistency (36)

The ranking table shown above summarizes the performance of classification algorithms with all feature selection methods by ranking them according to the t-test results. Cfs (25) and Gain (50) feature selection methods are the best suitable feature selection methods among classification algorithms. Bayesian, CART and J48 classification algorithms show good performance with Cfs (25) feature selection method as it can be seen from above ranking table. CART, J48 and Random forest classification algorithms give better performance with Gain (50) feature selection method. SMO classification algorithm has the best performance with reliefF (50) feature selection method.

Second research question is answered by using the t-test results. According to the results of t-test ranking, the compatibility of classification algorithms with feature selection algorithms is presented in table 4.15. According to this table, Bayesian algorithm is best compatible with Cfs (25) feature selection method. CART, J48 and Random Forest algorithms are best compatible with the Cfs (25) and Gain (50) feature selection methods. Moreover, SMO algorithm is compatible with the reliefF (50) feature selection method.

Table 4.15: The compatibility table for classification algorithms and feature selection methods

Classification Algorithm	Compatible Feature Selection Method
Bayesian	Cfs (25)
CART	Cfs (25) and Gain (50)
J48	Cfs (25) and Gain (50)
Random Forest	Gain (50) and Cfs (25)
SMO	ReliefF (50)

4.4 Evaluation of Clustering Analysis

In this section, the third research question about the usefulness of clustering analysis on the determination of the tendencies of permissions for the detection of malicious applications is studied. The implementation and the results of clustering analysis are explained. According to the results of clustering algorithm, the clusters are used to evaluate features for which they are mostly required by malicious applications.

K-means clustering algorithm is performed on the dataset-3784 which is the original dataset on hand for the analysis. K is chosen as 3 in the implementation of clustering analysis. K-means clustering algorithm divides the all apk files into 3 clusters. As a result of clustering implementation, the number of instances included in each cluster occurred as 597, 1,117 and 2,070 instances as given in below figure 4.10.

```
Clustered Instances

0      2070 ( 55%)
1      1117 ( 30%)
2       597 ( 16%)
```

Figure 4.10: The number of instances in each cluster

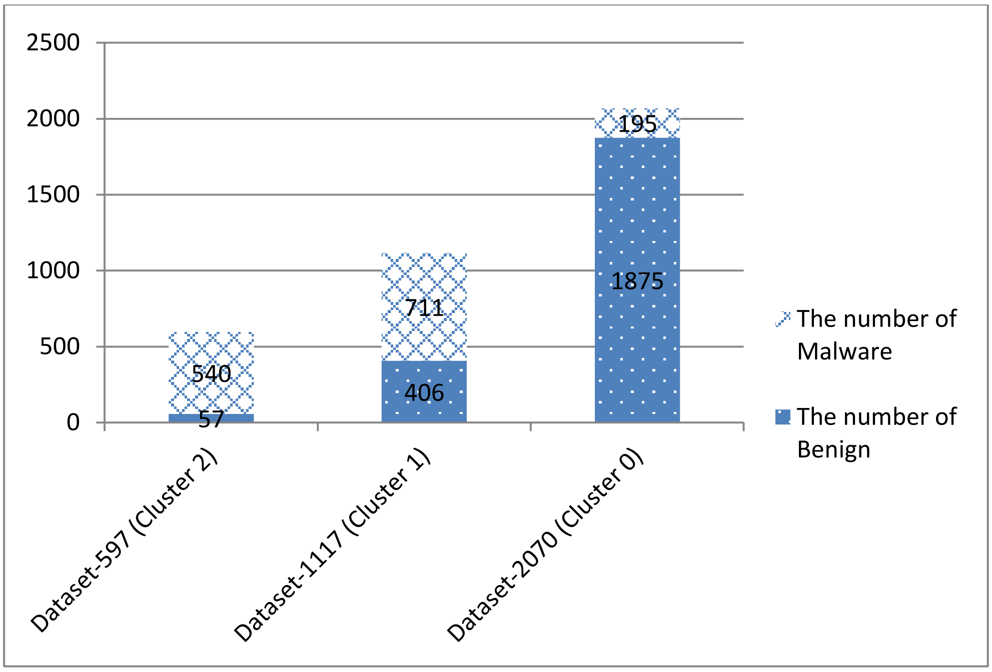


Figure 4.11: The number of benign and malware applications in clusters

The number of benign and malware applications in each cluster is shown in figure 4.11. According to the clustering results, cluster 0 contains mostly benign applications. Cluster 1 is mix of benign and malware applications. Cluster 2 consists of mostly malicious applications. The permissions are analyzed in this part of analysis according to their availability in each cluster. The centroid information given in the output of k-means clustering algorithm can be evaluated for which feature is effective on determining the benign or malware apk files. The centroid information for some features that is found as important in this study is given in table 4.16.

By making use of those cluster properties, the centroid information of clusters for each feature can be evaluated to determine the effect of permission on detecting the malicious apk files. The important features that are extracted from the clustering output are listed in table-4.16.

The centroid values for cluster-0, cluster-1 and cluster-2 can be seen in graph given in figure 4.12. The dots colored by green representing the cluster-0 are mostly located at the bottom of the graph. When we look at the number of benign and malware applications included in cluster-0, it can be said that this cluster is mostly formed by benign applications. This means that benign applications mostly do not require those permissions. The centroids of those permissions mostly below the 0.3, only Internet permission and Access Network State permission have larger centroid with 0.7072 and 0.4411 respectively.

Table 4.16: The centroids of some features for each cluster

	Full Data 3,784 instances	**Cluster-2 597 instances**	**Cluster-1 1,117 instances**	**Cluster-0 2,070 instances**
Weight	Mostly Benign	Mostly Malware	Benign-Malware	Mostly Benign
Benign	2,338 (% 61.79)	57 (% 9.55)	406 (% 36.35)	1,875 (% 90.58)
Malware	1,446 (% 38.21)	540 (% 90.45)	711 (% 63.65)	195 (% 9.42)
Access Network State	0.6607	0.8241	0.9803	0.4411
Change Network State	0.0996	0.5394	0.0412	0.0443
Access Wifi State	0.3449	0.6382	0.7377	0.0483
Change Wifi State	0.1712	0.5812	0.2534	0.0087
Internet	0.8362	0.9832	0.9964	0.7072
Read Phone State	0.5412	0.9866	0.9320	0.2019
Read SMS	0.1533	0.8827	0.0116	0.0193
Receive Boot Completed	0.2981	0.8375	0.4718	0.0488
Receive SMS	0.1913	0.9648	0.0439	0.0478
Send SMS	0.2196	0.9615	0,0671	0.0879
Write SMS	0.0740	0.4188	0.0090	0.0097
Wake Lock	0.3433	0.7638	0.5407	0.1155
Write External Storage	0.5185	0.9313	0.7878	0.2541
Mount Unmount Filesystems	0.1065	0.4322	0.1191	0.0058
Read Contacts	0.1607	0.7085	0.0645	0.0546
Restart Packages	0.1332	0.4456	0.1898	0.0126
Install Shortcut	0.1678	0.4154	0.3178	0.0155

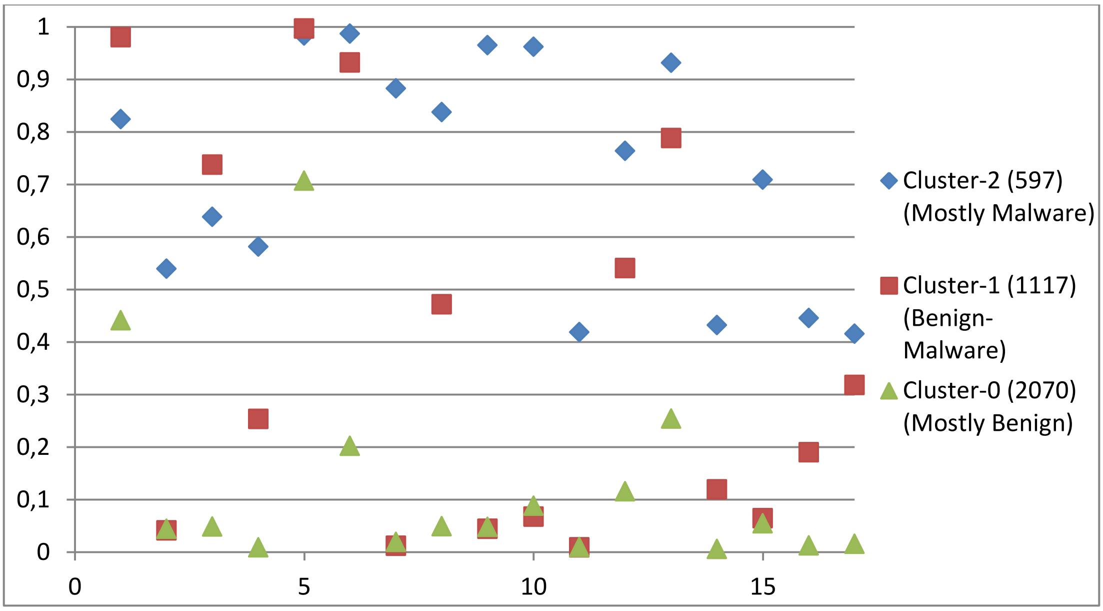


Figure 4.12: The scatter graph of centroids for clusters-0, cluster-1 and cluster-2

The dots colored by red in the graph show the centroid information in Cluster-1 which is dispersed to whole area on the graph. The number of benign and malware applications in this cluster gives us that this cluster is a mixture of benign and malware applications. The centroid information for permissions in this cluster changes between 0 and 1. The centroid information of cluster-1 is supporting our view about the effect of those permissions on detection of malicious apk files.

The dots colored by blue in the graph shows the centroid information in cluster-2 which are mostly located at the top of the scatter graph. Cluster-2 is mostly formed by malicious applications according to the number of benign and malware applications included in this cluster. Only, 4 permissions which are Write SMS, Mount Unmount Filesystems, Restart Packages and Install Shortcut are below the centroid level 0.5 but very close to 0.5. The remaining permissions have the centroid information above 0.5 and most of them are very close to 1.

Since the cluster-0 and cluster-2 are mostly formed by the benign and malware applications respectively. We could compare the centroid information of those clusters in order to determine the effect of permissions on the detection of malicious applications. For this purpose, cluster-0 and cluster-2 are compared to determine which features are existed in the malware weighted cluster.

In this way, Read Phone State, Read SMS, Receive Boot Completed, Receive SMS, Send SMS, Wake Lock, Write External Storage and Read Contacts features are strongly effective on determining the maliciousness of an apk file because the centroid information for those features are very close to 1 in cluster-2 meaning that those features are mostly permitted in the malicious apk files. The centroid information for those features in cluster-0 is very close to 0 meaning that those features are not mostly required in benign apk files. The centroid graph of those strong features in determining malicious applications is given in the Figure 4.13. In this figure, it can be seen that the centroids for cluster-2 is very close to 1 and the centroids for cluster-0 is very close to 0. For this reason, those 8 features are strongly effective on the detection of malware apk files. The function of those permissions can be seen from Developers (2014).

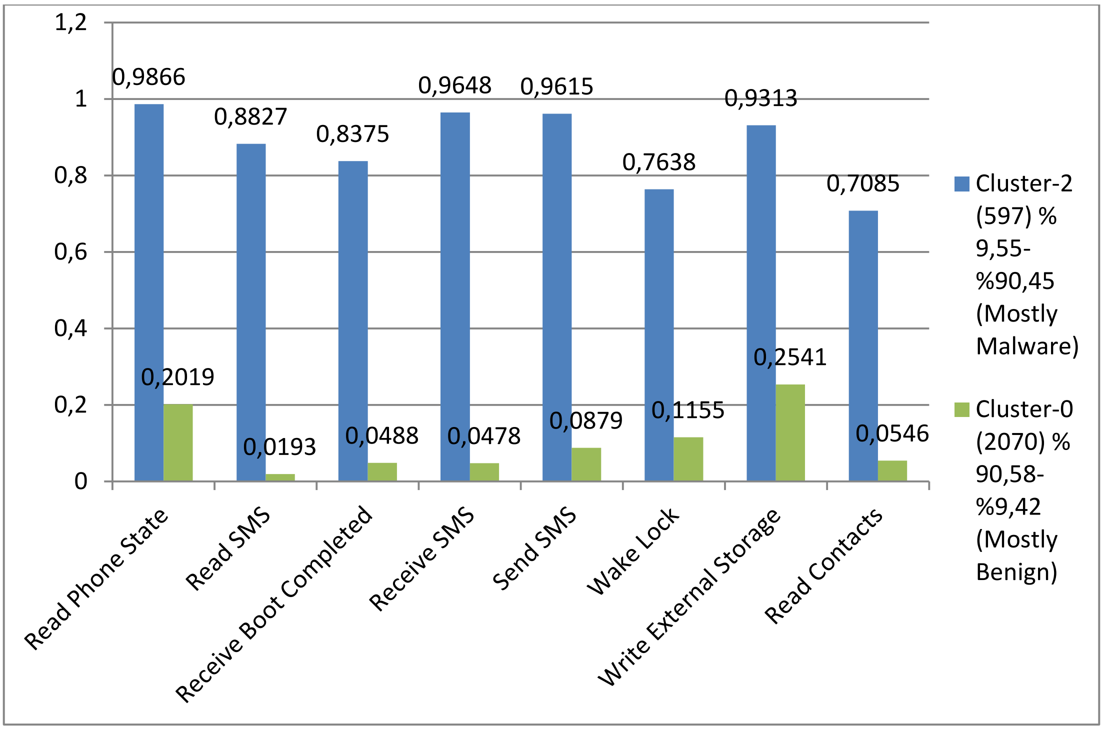


Figure 4.13: The centroids graph for strong features in determining malicious apk files

Access Network State, Change Network State, Access Wifi State, Change Wifi State, Write SMS, Mount Unmount Filesystems, Restart Packages and Install Shortcut features may become effective features on detecting malware apk files but they are not as strong as previous features given above. The graphical view of the centroids for those features that are defined as medium-strong feature for detecting malware apk files can be seen in below figure 4.14. For example, the centroid information of Access Network State in cluster-2 is 0.8241 which is showing strongness of this feature in determining malware apk files. However, the centroid of Access Network State feature in cluster-0 is 0.4411 which is reducing the strongness of this feature because this feature is also existed in the dataset including mostly benign apk files. Another example is the Change Network State feature. The centroid of this feature in cluster-2 is 0.5394 and the centroid of this feature in cluster-0 is 0.0443. This situation gives some clue in the tendency of this feature in the existence in malicious apk files but the strongness of this information is not much strong because the centroid of cluster-2 is 0.5394 which is not very close to 1. The same evaluation is also valid for other 6 features; Access Wifi State, Change Wifi State, Write SMS, Mount Unmount Filesystems, Restart Packages and Install Shortcut features.

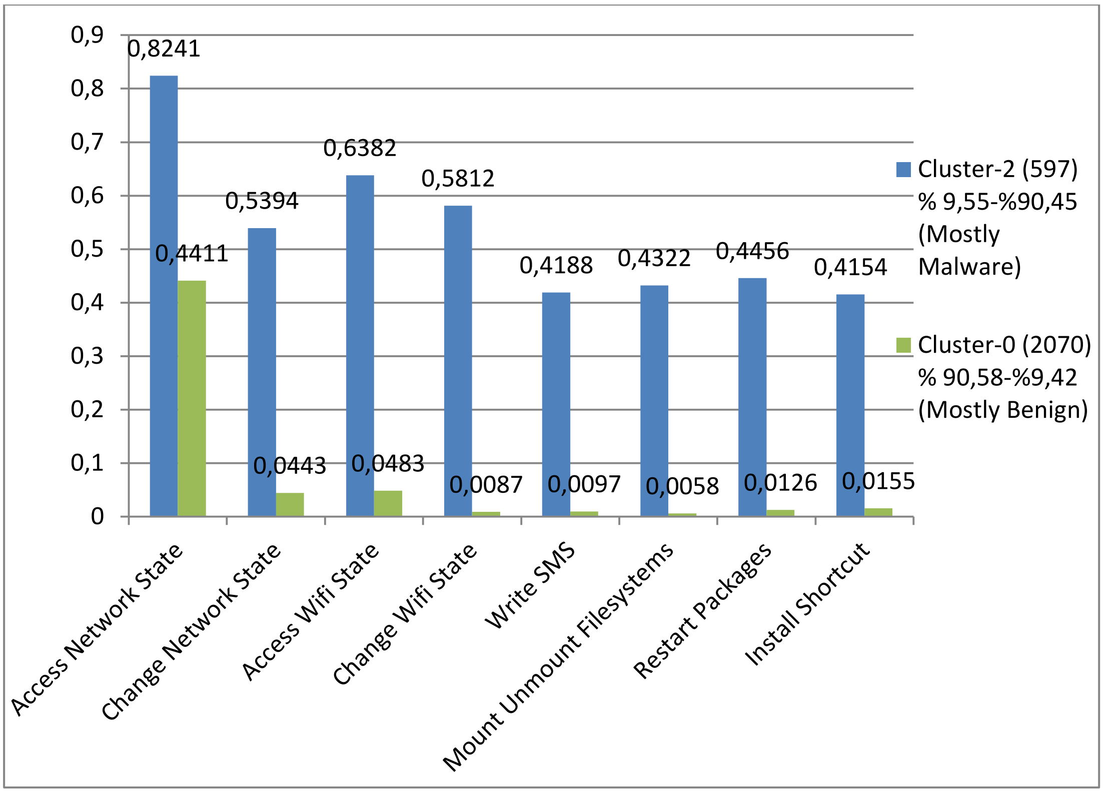


Figure 4.14: The centroids graph for medium-strong features in determining malicious apk files

The Internet feature seems to exist in two clusters; the centroid for internet feature in cluster-2 is 0.9832 and it is 0.7072 in cluster-0. By looking at this centroid values, we can say that internet feature is not important feature for detection of malicious apk files which is very reasonable because nowadays most of the android applications require internet permission. Other features excluding those 16 features are not effective on the detection of malware apk files according to the results of K-Means clustering algorithm implementation.

As a result of clustering analysis, the research question about clustering analysis is answered. The clustering analysis is beneficial to determine the permissions which are mostly required by malicious applications. The permissions are classified as strong and medium-strong permissions according to the results of clustering analysis. The permissions found as suspected can be further investigated in the application requiring those suspected permissions.

4.5 Evaluation of Dataset Size

In this section of the study, the last research question about the effect of dataset size on classification algorithms is tried to be answered. Four datasets are constructed from the dataset having 3,784 instances in order to implement classification algorithms and evaluate their performances on those datasets. The evaluation of implementation results is given in this section in detailed.

4.5.1 Formation of Datasets

The last aim of this study is to measure the effect of dataset size on the detection of malicious apk files. It is investigated that in which dataset size, the analysis gives the best malware detection results. Increasing the dataset size is whether relevant or not better detecting malicious apk files. For this reason, it is needed to generate new datasets from the dataset-3784 which is on hand. 3 more datasets are generated by randomly selecting apk files. The dataset sizes are chosen as 500, 1,000, 2,500 and 3,784. The construction process of datasets is displayed by a graph in figure 4.15. Moreover, the number of benign apk files and malicious apk files in all datasets are chosen as equivalent as it is shown in figure 4.16. 4 datasets are dependent each other because some of the apk files are within them.

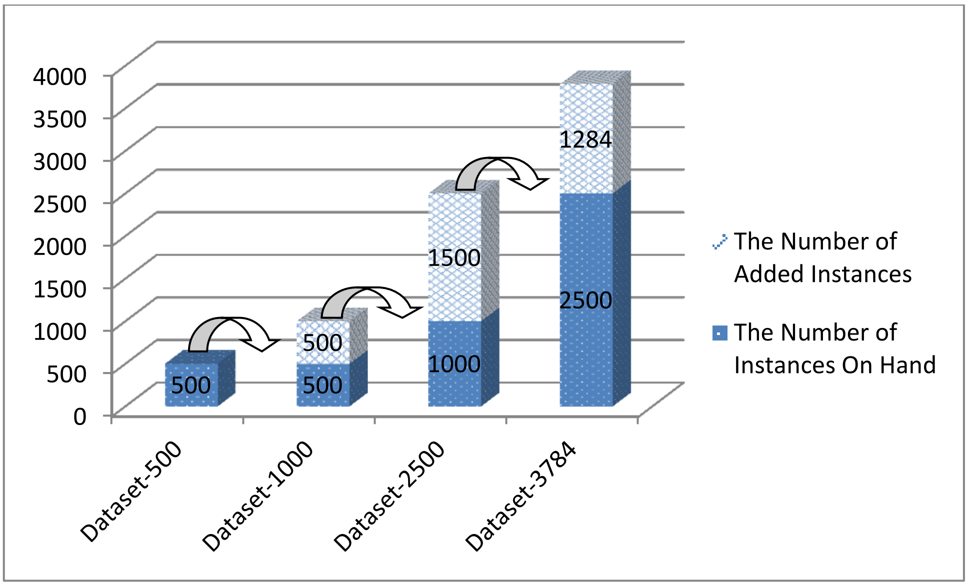


Figure 4.15: The graphical representation of formation of datasets

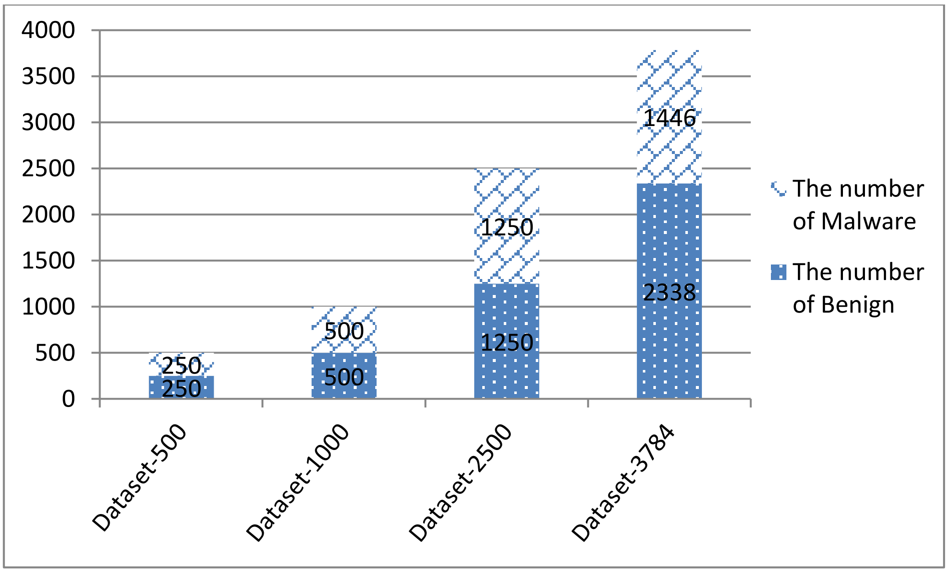


Figure 4.16: The number of benign and malware applications in the datasets

4.5.2 Feature Selection Implementation and Evaluation

The best performing two classification algorithms which are Random forest and J48 decision tree are used for the classification analysis on the constructed four datasets. The most compatible feature selection method with those classification algorithms is Cfs subset feature selection method with 25 features as it can be seen from table 4.15. Cfs subset feature selection method is implemented in the selection of features for four datasets formed in previous section namely dataset-500, dataset-1000, dataset-2500 and dataset-3784. Table that summarizing the output of feature selection method implementations is given in the Appendix-C. According to the results of Cfs subset feature selection method implementation, totally 28 features are selected in 4 datasets. And 21 of those 28 features are selected in all 4 datasets as shown in figure 4.17.

“android.permission.BROADCAST_STICKY”,
“android.permission.DELETE_CACHE_FILES” permissions are only selected in feature selection of dataset-500.

“android.permission.CHANGE_NETWORK_STATE” permission is selected in two feature selection implementations which are dataset-500 and dataset-1000.

“android.permission.DELETE_PACKAGES”,
“android.permission.INSTALL_PACKAGES”,
“android.permission.RESTART_PACKAGES”
“android.permission.WRITE_SETTINGS”

permissions are selected in three feature selection implementations. As a result, 21 features are found important in the implementation of feature selection method in four datasets. The list of those features can be seen in the Appendix-C. The implementation of feature selection in dataset-2500 and dataset-3784 gives the same result with same features. Dataset-500 and dataset-1000 shows slightly different results from dataset-2500 and dataset-3784. When we look at the feature selection implementation results, the selected features do not show much variety in four different datasets which means that the number of instances in the dataset is not much effective on the feature selection method implementation. Another reason of this result is the dependency of datasets formed by randomly adding new instances. Datasets are covering other datasets having lower number of instances.

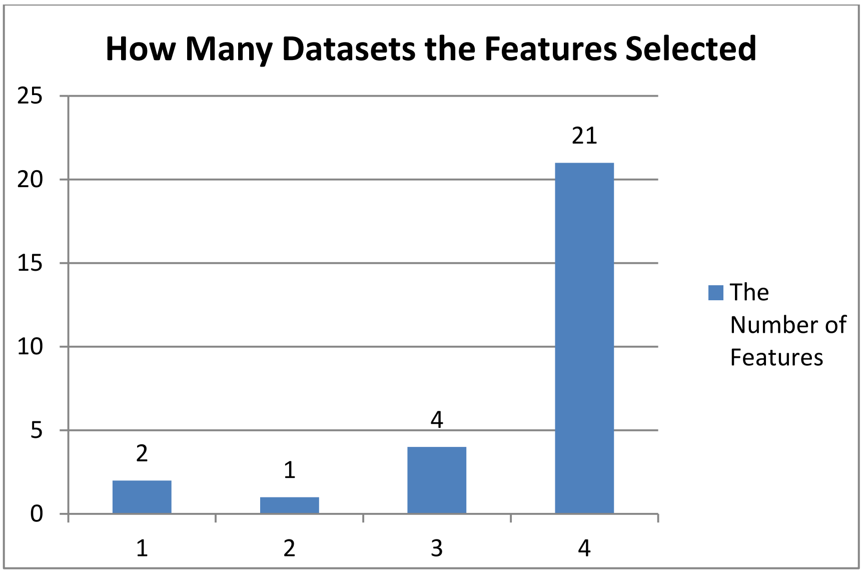


Figure 4.17: The number of features selected in how many datasets

By implementing the feature selection method, we have the ability of reducing the one dimension of the dataset that is we are reducing the horizontal side of data matrix from 182 features to 25 features. In this way, we are reducing approximately % 86 of whole dataset by keeping its original meaning as much as possible. The vertical dimension of the dataset does not change after the feature selection method implementation.

4.5.3 Classification Algorithm Implementation and Evaluation

Random forest (RF) and J48 decision tree algorithms are made use of in evaluation of dataset sizes. The reason of selecting two classification algorithms is to see the performance of different classification algorithms with datasets having different number of instances.

The datasets in which, the feature dimension is reduced in previous section by implementing feature selection method are made use of in the implementation of classification algorithms. We have two classification algorithms hence totally we have eight iterations for the classification algorithm implementation. Weka tool is utilized in the implementation of classification algorithms. The results of the classification algorithms implementation are summarized in table 4.17.

Table 4.17: The classification results for different dataset sizes

Data Amount	Classification Algorithm	Overall Accuracy	TP Rate	FP Rate	Precision
Dataset-500	Random Forest	92.20%	0.922	0.078	0.922
Dataset-500	J48 Decision Tree	90.40%	0.904	0.096	0.904
Dataset-1000	Random Forest	93.40%	0.934	0.066	0.934
Dataset-1000	J48 Decision Tree	92.30%	0.923	0.077	0.923
Dataset-2500	Random Forest	93.88%	0.939	0.061	0.939
Dataset-2500	J48 Decision Tree	93.04%	0.930	0.070	0.930
Dataset-3784	Random Forest	94.90%	0.949	0.060	0.949
Dataset-3784	J48 Decision Tree	93.87%	0.939	0.072	0.939

All classification algorithm results are over the 92 % in all datasets. Random forest algorithm gives the better results than J48 decision tree algorithm in all datasets. Both classification algorithms give better results with the increase in dataset size as shown in figure 4.18. This means that the datasets having more than 3,784 instances might give better performance than those results obtained in this study.

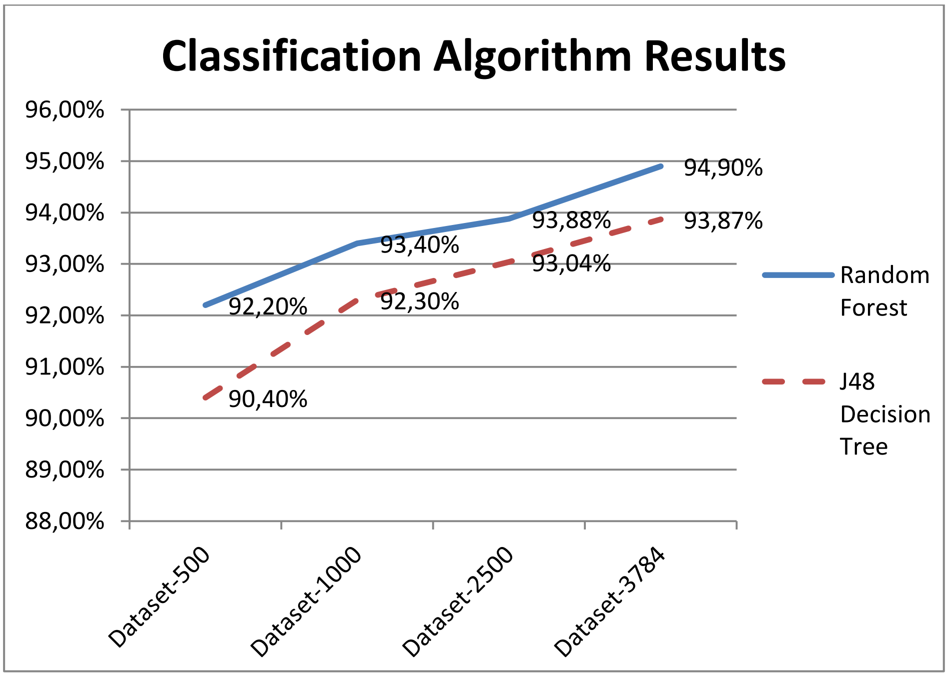


Figure 4.18: The graph of Random Forest and J48 Decision Tree algorithm results among different dataset sizes

4.6 Summary of Findings

In chapter-4, the answers for four research questions defined at the beginning of this study are tried to be found by evaluating the results of analysis. The first question is about finding the best performing classification algorithm among Bayesian, Classification and Regression Tree (CART), J48 Decision Tree, Random Forest and Sequential Minimal Optimization (SMO). To answer this question, the classification algorithms are implemented on the datasets in which the features are selected by using four different feature selection methods. 35 combinations of classification algorithm, feature selection method and feature number is implemented in order to find answer first question. The results of classification algorithms are evaluated by paired t-test with their prediction accuracies. As a result of evaluations, random forest algorithm is found as best performing algorithm. Moreover, J48 decision tree algorithm is evaluated as having second best performance between classification algorithms. Second question of this study is to find answer for the compatible classification algorithms and feature selection methods. For this purpose the implementation results of 35 combinations of classification algorithms, feature selection methods and feature number is made use of. Again, paired t-test performed in order to compare the results statistically. According to the comparison results, Bayesian is best compatible with Cfs subset feature selection method with 25 features. CART, J48 and Random Forest are best compatible with Cfs subset feature selection method with 25 features and Gain Ratio attribute feature selection method with 50 features. SMO shows best compatibility with ReliefF attribute feature selection method with 50 features. The answer for third question is found by applying k-means clustering algorithm and evaluating the permissions in each cluster. According to the evaluation results, the clustering analysis is found important for determining the permissions mostly required by malicious applications. 8 permissions are found as strong permissions which are mostly required by malicious applications. Moreover, 8 permissions are found medium-strong permissions which are found mostly malicious applications but not as much as first 8 permissions. The last question of this study is about the effect of dataset size on the performance of detecting malicious applications. For this purpose, the datasets with 500, 1,000, 2,500 and 3,784 instances are used for the implementation of Random Forest and J48 decision tree classification algorithms with Cfs subset feature selection method. According to the results, the prediction accuracy of both classification algorithms is increasing with the increase in the dataset size. This means that the dataset having more than 3,784 instances might give even better results.

CHAPTER V

5 CONCLUSION AND IMPLICATIONS

5.1 Discussion and Conclusion

The detection of malicious activities is an important problem in the applications used in the mobile devices. The usage of mobile devices has been increased very fast in past decade. And the increase trend seems to continue in the following decade at least same speed. The popularity of mobile devices attracts the attackers to benefit from the people who are using mobile device. Because those mobile devices include very important information of people and people uses those devices in their important tasks such as banking, health, communication. Bad activities encountered in mobile devices can be listed as novelty and amusement, selling user information, premium-rate calls and SMS, SMS spam, search engine optimization and ransom. All those activities give benefits to attacker especially as financial gains and give damage to the users of the mobile devices especially as financial loss.

In order to prevent the illegal activities performed by bad guys, the activities made in mobile devices are investigated for the detection of unwanted activities. For this purpose, the applications installed on mobile devices are the most important factor in illegal activities. The detection of malicious applications is carried out by analyzing the information of those mobile applications and the results of activities performed by those applications. The way of detection methods is split into two ways as static analysis which is the analysis performed without running the applications and dynamic analysis which is performed by run-time monitoring. In this study, static analysis is preferred by investigating the permission required by mobile applications in order to use some resources in the mobile devices. The permissions are included in the installation packet of mobile applications and the permissions must be accepted by user of mobile device in order to install that application on mobile device. The permissions required for mobile applications were studied before in different studies. However, the number of permissions that can be asked by mobile devices has been expanding continuously. The permission list published latest is API Level-20. This means that until now 19 permission lists were published by android. In this study, the latest published permission list is made use in order to include latest presented permissions on the detection of malicious applications. The android applications used in this study are taken from COMODO laboratory in METU. The status of applications as benign or malware is also given by COMODO. This information is reliable because the employees of COMODO work on the code of applications one by one in order to determine whether related application is malware or not. Dataset including 3,784 android applications is gathered from COMODO. The permissions within those android applications are searched in order to build dataset which is used in this study.

In this study, four research questions are tried to be answered with static analysis by using permission data of android applications. In the first research question, the classification algorithms are questioned in order to find best performing classification algorithm in the detection of malicious applications in android applications. In previous studies, classification algorithms were used within the prediction models to detect malicious applications. Bayesian algorithm and some decision tree algorithms such as CART, J48 and Random Forest were used in different studies with different datasets. (Yerima, et al., 2013) and (Zaw & Aung, 2013) There was no study that used support vector machine (SVM) algorithm in the detection of malicious applications. In this study, classification algorithms from different classification category namely; bayesian, decision tree and Support Vector Machine (SVM) are chosen in order to evaluate their performance on same dataset. Bayesian, CART, J48, Random Forest and SMO classification algorithms are implemented in this study. The implementation of classification algorithms are realized on the datasets in which four different feature selection methods are implemented. Therefore, the comparison of classification algorithms are performed with different feature selection methods. Paired t-test is used in order to compare prediction accuracies of the classification algorithms. According to the results of t-test, Random Forest algorithm and J48 decision tree algorithm are found best performing classification algorithms. In addition, decision tree algorithms are better performing than the SVM and bayesian algorithms.

In second research question, the compatibility of classification algorithms with feature selection methods is searched. In the literature, there was no study that is questioning the most suitable feature selection method for classification algorithms. However, there are studies using a selected feature selection method without questioning its performance. (Yerima, et al., 2013) and (Zaw & Aung, 2013) In order to make comparison of classification algorithms with feature selection methods, 35 combinations of classification algorithms, feature selection methods and feature number are implemented. The prediction accuracy of classification algorithms with all feature selection methods are compared with t-test results. According to the results, Bayesian and Cfs subset with 25 features; CART, J48, Random Forest, and Cfs subset with 25 features and Gain Ratio with 50 features; and SMO and reliefF attribute with 50 features are found as best performing classification algorithm and feature selection matches.

In third research question, the usage of clustering algorithm in the detection of malicious applications is asked. K-means clustering algorithm is used in two studies in order to detect malicious applications. (Abu Samra, et al., 2013), (Zaw & Aung, 2013) However, there is no evaluation on the permissions within clusters after applying clustering algorithm in those studies. In this study, it is aimed to find permissions that are mostly wanted by malicious applications. In this way, an application requiring found permissions are carefully examined further whether it is malware or not. 3 clusters are constructed by applying k-means clustering algorithm. According to the results of clustering, the characteristic of three clusters are formed by benign application weighted, malware application weighted and benign-malware application mixed clusters. The evaluation of permission data for applications is made according to the characteristics of clusters. In this way, the permissions that are evaluated as the sign for the malicious application are determined as strong permissions and medium-strong permissions.

In the last research question, it is examined that the dataset size is effective on the performance of classification algorithms. In the literature, there is no study questioning the dataset size effect on the performance of classification algorithms for the detection of malicious applications. In this study, the datasets including 500, 1,000, 2,500 and 3,784 instances are constructed to analyze the dataset size effect. Random Forest and J48 decision tree classification algorithms are used with Cfs subset features selection method with 25 features. According to the results, the prediction accuracy of classification algorithms increases with larger datasets. In addition, Random Forest algorithm shows better performance than J48 decision tree algorithm in those implementations.

5.2 Study Limitations

This study uses some of the well-known methods and algorithms in data mining. The results of analysis strictly depend on the theories behind those methods and algorithms. In addition, the implementation of those methods and algorithms are made by using Weka Machine Learning Tool. The implementation results are limited to the performance of Weka tool as well.

The analysis made in this study is limited to the dataset used in the analysis which is provided from COMODO Security Solutions Company. The datasets including different applications may give different results in the analysis. In addition, 3.784 data is included in the dataset. The analysis is limited to dataset size as well.

In this study, only static analysis is performed by using permission data which are included in the installation packet of android applications. The dynamic analysis can be performed by using run-time data of android applications as well.

5.3 Implications

The analyzer working over malware detection may use the methods and algorithms proposed as a result of this study. The success of malware detection is strongly depended on the used methods and algorithms in the detection analysis. Moreover, other feature selection methods and classification algorithms may be analyzed by the researchers in order to expand usable methods and algorithms in the detection of malware.

In this study, the permission data is made use for the detection analysis of malware in mobile applications. Some other data which are included in the installation packet of mobile applications can be used in the analysis of malicious applications.

In this study, static analysis is performed by using permission data included in the installation packet of android applications which are not run-time data of applications. Further analysis can be performed by the analyzers by using dynamic analysis which is performed by using run-time data of mobile applications.

In this study, the dataset including 3.784 data is made use of in the analysis. The dataset sizes below 3.784 are constructed by using this dataset in order to measure the effect of dataset size. The larger datasets may become more useful in further studies to measure the effect of dataset size in malware detection analysis.

6 REFERENCES

Abu Samra, A. A., Yim, K. & Ghanem, O. A., 2013. Analysis of Clustering Technique in Android Malware Detection. *IEEE.*

Amamra, A., Talhi, C. & Robert, J. M., 2012. Smartphone Malware Detection: From a Survey Towards Taxonomy. *IEEE.*

Amos, B., Turner, H. & White, J., 2013. Applying machine learning classifiers to dynamic Android malware deection at scale. *IEEE.*

2013. Mobile Security Guide: Protect Your Organization From Mobile Malware. *Rapid7 Corporate Headquarters.*

2014. *Classification and Regression Trees (C$RT).* [Online] Available at: www.obgyn.cam.ac.uk/cam-only/statsbook [Accessed 2014].

Baksmali. [Online] Available at: http://code.google.com/p/smali [Accessed 2014].

Apvrille, A. & Strazzere, T., 2012. Reducing the window of opportunity for Android malware Gotta catch 'em all. *Journal in Computer Virology,* pp. vol. 8, pp. 61-71.

Breiman, L., 2001. Random Forests. Machine Learning.. pp. 45 (1): 5-32.

Breiman, L. et al., 1998. *Classification and Regression Trees.* California: Belmont.

Chen, E., 2011. *What are the advantages of different classification algorithms?.* [Online] Available at: http://www.quora.com/What-are-the-advantages-of-different-classification-algorithms [Accessed September 2014].

COMODO Group, Inc., 2014. *COMODO.* [Online] Available at: https://tr.comodo.com/ [Accessed 2014].

Developers, 2014. *App Manifest.* [Online] Available at: http://developer.android.com/guide/topics/manifest/manifest-intro.html [Accessed 2014].

Developers, 2014. *Manifest.permission.* [Online] Available at: http://developer.android.com/reference/android/Manifest.permission.html [Accessed 2014].

Devesa, J. et al., 2010. Automatic Behaviour-based Analysis and Classification System for Malware Detection. *Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS),* pp. 395-399.

Felt, P. et al., 2011. A Survey of Mobile Malware in the Wild.

Gartner, 2014. *Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013.* [Online] Available at: http://www.gartner.com/newsroom/id/2665715 [Accessed 2014].

George, H. J. & Langley, P., 1995. Estimating Continuous Distributions in Bayesian Classifiers. *Eleventh Conference on Uncertainty in Artificial Intelligence,* pp. 338-345.

Google, 2014. *android-apktool, A tool for reverse engineering Android apk files.* [Online] Available at: http://code.google.com/p/android-apktool [Accessed 2014].

Guszcza, J., 2005. *CART from A to B,* Chicago: Deloitte.

Hall, M. A., 1998. *Correlation-based Feature Subset Selection for Machine Learning.* New Zealand: Hamilton.

Han, J. & Kamber, M., 2000. *Data Mining: Concepts and Techniques.* s.l.:Morgan Kaufmann Publishers.

Joachims, T., 1998. *Making Large-Scale SVM Learning Practical,* Dortmund: UNIDO.

Keller, F., n.d. *Connectionist and Statistical Language Processing, Evaluation,* s.l.: s.n.

Kononenko, I., 1994. Estimating Attributes: Analysis and Extensions of RELIEF. *European Conference on Machine Learning,* pp. 171-182.

Liu, H. & Setiono, R., 1996. A probabilistic approach to feature selection - A filter solution. *13th International Conference on Machine Learning,* pp. 319-327.

Mohata, V., Dakhane, D. M. & Pardhi, R. L., 2013. Mobile Malware detection Techniques. *IJCSET.*

Platt, J., 1998. Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in Kernel Methods - Support Vector Learning.*

Quinlan, R., 1993. *C4.5: Programs for Machine Learning.* CA.: San Mateo.

Santos, I., Brezo, F., Ugarte-Pedrero, X. & Bringas, P. G., 2011. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences.*

Santos, I., Laorden, C. & Bringas, G., 2011. Collective classification for unknown malware detection. *Proceedings of the 6th International Conference on Security and Cryptography (SECRYPT),* pp. 251-256.

Santos, I., Nieves, J. & Bringas, P. G., 2011. Semi-supervised learning for unknown malware detection. *Proceedings of the 4th International Symposium on Distributed Computing and Artificial Intelligence (DCAI). 9th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS),* pp. 415-422.

Schultz, M., Eskin, E., Zadok, F. & Stolfo, S., 2001. Data mining methods for detection of new malicious executables. *Proceedings of the 2001 IEEE Symposium on Security and Privacy,* pp. 38-49.

Statista, 2014. *The number of cumulative app downloads from the Google Play app store for Android devices between August 2010 to July 2013,.* [Online] Available at: http://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/ [Accessed 2014].

Suarez-Tangil, G., Tapiador, E., Peris-Lopez, P. & Blasco, J., 2014. DENDROID: A text mining approach to analyzing and classifying code. *Elsevier*, pp. 1104-1117.

Sudheer, M. et al., 2013. A combitorial Approach for New Generic Malware Detection Technique. *The International Journal of Computer Science & Applications (TIJCSA).*

Thomas, Z. et al., 2013. Power Consumption-based Application Classification and Malware Detection on Android Using Machine-Learning Techniques. *IARIA.*

Thomson, I., 2012. *Google celebrates as Android hits 25 billion downloads.* [Online]Availableat:http://www.theregister.co.uk/2012/09/26/google_play_25bn_downloads/ [Accessed 2014].

Wikipedia, 2013. *Bayes Classifier*. [Online] Available at: http://en.wikipedia.org/wiki/Bayes_classifier [Accessed 2014].

Witten, I. H., Frank, E. & Hall, M. A., 2011. *Data Mining Practical Machine Learning Tools and Techniques.* s.l.:Morgan Kaufmann Publishers.

Yerima, S. Y., Sezer, S., McWilliams, G. & Muttik, I., 2013. A New Android Malware Detection Approach Using Bayesian Classification. *IEEE.*

Zaw, W. & Aung, Z., 2013. Permission-Based Android Malware Detection. *IJSTR.*

Zhao, Y. & Zhang, Y., 2007. Comparison of decision tree methods for finding active objects. *Advances of Space Research.*

Zhu, X. & Peiravian, N., 2013. Machine Learning for Android Malware Detection Using Permission and API Calls. *IEEE 25th International Conference on Tools with Artificial Intelligence..*

Thomson, Iain, 2012. Google celebrates as Android hits 25 billion downloads.[Online]Availableat:http://www.theregister.co.uk/2012/09/26/google_play_25bn_downloads/ [Accessed 2014]

7 APPENDICES

APPENDIX-A: The Permissions listed in API Level 20

- android.permission.ACCES_MOCK_LOCATION
- android.permission.ACCESS_ASSISTED_GPS
- android.permission.ACCESS_CELL_ID
- android.permission.ACCESS_CHECKIN_PROPERTIES
- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_COARSE_UPDATES
- android.permission.ACCESS_COURSE_LOCATION
- android.permission.ACCESS_DRM
- android.permission.ACCESS_FIND_LOCATION
- android.permission.ACCESS_FINE_LOCATION
- android.permission.ACCESS_GPS
- android.permission.ACCESS_LOCATION
- android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
- android.permission.ACCESS_MOCK_LOCATION
- android.permission.ACCESS_NETWORK_STATE
- android.permission.ACCESS_SURFACE_FLINGER
- android.permission.ACCESS_WIFI_STATE
- android.permission.ACCOUNT_MANAGER
- android.permission.ADD_SYSTEM_SERVICE
- android.permission.AUTHENTICATE_ACCOUNTS
- android.permission.BAIDU_LOCATION_SERVICE
- android.permission.BATTERY_STATS
- android.permission.BIND_ACCESSIBILITY_SERVICE
- android.permission.BIND_APPWIDGET
- android.permission.BIND_DEVICE_ADMIN
- android.permission.BIND_INPUT_METHOD
- android.permission.BIND_NFC_SERVICE
- android.permission.BIND_NOTIFICATION_LISTENER_SERVICE
- android.permission.BIND_PRINT_SERVICE
- android.permission.BIND_REMOTEVIEWS
- android.permission.BIND_TEXT_SERVICE
- android.permission.BIND_VPN_SERVICE
- android.permission.BIND_WALLPAPER
- android.permission.BLUETOOTH
- android.permission.BLUETOOTH_ADMIN
- android.permission.BLUETOOTH_PRIVILEGED
- android.permission.BRICK
- android.permission.BROADCAST_PACKAGE_REMOVED
- android.permission.BROADCAST_SMS
- android.permission.BROADCAST_STICKY
- android.permission.BROADCAST_WAP_PUSH
- android.permission.CALL_PHONE
- android.permission.CALL_PRIVILEGED
- android.permission.CAMERA
- android.permission.CAPTURE_AUDIO_OUTPUT
- android.permission.CAPTURE_SECURE_VIDEO_OUTPUT
- android.permission.CAPTURE_VIDEO_OUTPUT
- android.permission.CHANGE_COMPONENT_ENABLED_STATE
- android.permission.CHANGE_CONFIGURATION
- android.permission.CHANGE_NETWORK_STATE
- android.permission.CHANGE_WIFI_MULTICAST_STATE
- android.permission.CHANGE_WIFI_STATE
- android.permission.CLEAR_APP_CACHE
- android.permission.CLEAR_APP_USER_DATA
- android.permission.CONTROL_LOCATION_UPDATES
- android.permission.DELETE_CACHE_FILES
- android.permission.DELETE_PACKAGES
- android.permission.DUMP
- android.permission.EXPAND_STATUS_BAR
- android.permission.FACTORY_TEST
- android.permission.FLASHLIGHT
- android.permission.FORCE_BACK
- android.permission.FORCE_STOP_PACKAGES
- android.permission.FULLSCREEN
- android.permission.GET_ACCOUNTS
- android.permission.GET_PACKAGE_SIZE
- android.permission.GET_TASKS
- android.permission.GET_TOP_ACTIVITY_INFO
- android.permission.GLOBAL_SEARCH
- android.permission.HARDWARE_TEST
- android.permission.INJECT_EVENTS
- android.permission.INSTALL_LOCATION_PROVIDER
- android.permission.INSTALL_PACKAGES
- android.permission.INTERNAL_SYSTEM_WINDOW
- android.permission.INTERNET
- android.permission.KILL_BACKGROUND_PROCESSES
- android.permission.LOCATION_HARDWARE
- android.permission.MANAGE_ACCOUNTS
- android.permission.MANAGE_APP_TOKENS
- android.permission.MANAGE_DOCUMENTS
- android.permission.MASTER_CLEAR
- android.permission.MEDIA_CONTENT_CONTROL
- android.permission.MODIFY_AUDIO_SETTINGS
- android.permission.MODIFY_PHONE_STATE
- android.permission.MOUNT_FORMAT_FILESYSTEMS
- android.permission.MOUNT_UNMOUNT_FILESYSTEMS
- android.permission.NFC
- android.permission.PERMISSION_NAME
- android.permission.PERSISTENT_ACTIVITY
- android.permission.PROCESS_INCOMING_CALLS
- android.permission.PROCESS_OUTGOING_CALLS
- android.permission.QUERY_DRM
- android.permission.RAISED_THREAD_PRIORITY
- android.permission.READ_APN_SETTINGS
- android.permission.READ_CALENDAR
- android.permission.READ_CALL_LOG
- android.permission.READ_CONTACTS
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.READ_FRAME_BUFFER
- android.permission.READ_INPUT_STATE
- android.permission.READ_LOGS
- android.permission.READ_OWNER_DATA
- android.permission.READ_PHONE_STATE
- android.permission.READ_PROFILE
- android.permission.READ_SECURE_SETTINGS
- android.permission.READ_SETTINGS
- android.permission.READ_SMS
- android.permission.READ_SOCIAL_STREAM
- android.permission.READ_SYNC_SETTINGS
- android.permission.READ_SYNC_STATS
- android.permission.READ_USER_DICTIONARY
- android.permission.REBOOT
- android.permission.RECEIVE_BOOT_COMPLETED
- android.permission.RECEIVE_MMS
- android.permission.RECORD_AUDIO
- android.permission.RECORD_VIDEO
- android.permission.REORDER_TASKS
- android.permission.RESTART_PACKAGES
- android.permission.SEND_RESPOND_VIA_MESSAGE
- android.permission.SEND_SMS
- android.permission.SET_ACTIVITY_WATCHER
- android.permission.SET_ALWAYS_FINISH
- android.permission.SET_ANIMATION_SCALE
- android.permission.SET_DEBUG_APP
- android.permission.SET_ORIENTATION
- android.permission.SET_POINTER_SPEED
- android.permission.SET_PREFERRED_APPLICATIONS
- android.permission.SET_PROCESS_LIMIT
- android.permission.SET_TIME
- android.permission.SET_TIME_ZONE
- android.permission.SET_WALLPAPER
- android.permission.SET_WALLPAPER_HINTS
- android.permission.SIGNAL_PERSISTENT_PROCESSES
- android.permission.STATUS_BAR
- android.permission.SUBSCRIBED_FEEDS_READ
- android.permission.SUBSCRIBED_FEEDS_WRITE
- android.permission.SYSTEM_ALERT_WINDOW
- android.permission.TRANSMIT_IR
- android.permission.UPDATE_DEVICE_STATS
- android.permission.USE_CREDENTIALS
- android.permission.USE_SIP
- android.permission.VIBRATE
- android.permission.WAKE_LOCK
- android.permission.WRITE_APN_SETTINGS
- android.permission.WRITE_CALENDAR
- android.permission.WRITE_CALL_LOG
- android.permission.WRITE_CONTACTS
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.WRITE_GSERVICES
- android.permission.WRITE_MEDIA_STORAGE
- android.permission.WRITE_OWNER_DATA
- android.permission.WRITE_PROFILE
- android.permission.WRITE_SECURE_SETTINGS
- android.permission.WRITE_SETTINGS
- android.permission.WRITE_SMS
- android.permission.WRITE_SOCIAL_STREAM
- android.permission.WRITE_SYNC_SETTINGS
- android.permission.WRITE_USER_DICTIONARY
- archos.permission.FULLSCREEN.FULL
- com.android.alarm.permission.SET_ALARM
- com.android.browser.permission.READ_HISTORY_BOOKMARKS
- com.android.browser.permission.WRITE_HISTORY_BOOKMARKS
- com.android.launcher.permission.INSTALL_SHORTCUT
- com.android.launcher.permission.UNINSTALL_SHORTCUT
- com.android.voicemail.permission.ADD_VOICEMAIL
- phone.android.setting.SENDFILES
- phone.android.setting.SERVICESTART
- phone.android.setting.SERVICESTOP
- phone.android.setting.SETHOME
- phone.android.setting.START_RECORDING
- phone.android.setting.STARTTRIP

android.permission.DEVICE_POWER	android.permission.RECEIVE_SMS	phone.android.setting.STOP_RECORDING
android.permission.DIAGNOSTIC	android.permission.RECEIVE_USER_PRESENT	phone.android.setting.STOPTRIP
android.permission.DISABLE_KEYGUARD	android.permission.RECEIVE_WAP_PUSH	phone.android.setting.TRIPOVERRIDE

APPENDIX-B: The Results for the Combinations of Classification Algorithms and Feature Selection Methods

Feature Selection Method	Classification Algorithm	Overall Accuracy (ACC)	TP Rate	FP Rate	Precision
Cfs Subset (25)	Bayesian	89.32%	0.893	0.120	0.893
Cfs Subset (25)	CART	93.58%	0.936	0.073	0.936
Cfs Subset (25)	J48	93.87%	0.939	0.072	0.939
Cfs Subset (25)	Random Forest	94.90%	0.949	0.060	0.949
Cfs Subset (25)	SMO	91.62%	0.916	0.082	0.918
Cfs Subset (50)	Bayesian	88.35%	0.883	0.139	0.883
Cfs Subset (50)	CART	89.98%	0.900	0.097	0.903
Cfs Subset (50)	J48	90.51%	0.905	0.078	0.913
Cfs Subset (50)	Random Forest	92.79%	0.928	0.078	0.928
Cfs Subset (50)	SMO	92.68%	0.927	0.082	0.927
Consistency Subset (36)	Bayesian	72.89%	0.729	0.367	0.727
Consistency Subset (36)	CART	74.05%	0.740	0.373	0.749
Consistency Subset (36)	J48	73.39%	0.734	0.384	0.744
Consistency Subset (36)	Random Forest	74.21%	0.742	0.369	0.750
Consistency Subset (36)	SMO	72.78%	0.728	0.399	0.742
Gain Ratio Attribute (25)	Bayesian	87.66%	0.877	0.150	0.876
Gain Ratio Attribute (25)	CART	91.99%	0.920	0.081	0.921
Gain Ratio Attribute (25)	J48	92.23%	0.922	0.078	0.924
Gain Ratio Attribute (25)	Random Forest	92.15%	0.922	0.081	0.922
Gain Ratio Attribute (25)	SMO	91.15%	0.911	0.085	0.914
Gain Ratio Attribute (50)	Bayesian	87.95%	0.879	0.146	0.879
Gain Ratio Attribute (50)	CART	93.60%	0.936	0.071	0.936
Gain Ratio Attribute (50)	J48	93.90%	0.939	0.069	0.939
Gain Ratio Attribute (50)	Random Forest	94.50%	0.945	0.062	0.945
Gain Ratio Attribute (50)	SMO	92.60%	0.926	0.081	0.926
ReliefF Attribute (25)	Bayesian	87.18%	0.872	0.155	0.871
ReliefF Attribute (25)	CART	89.59%	0.896	0.103	0.899
ReliefF Attribute (25)	J48	90.35%	0.904	0.085	0.909
ReliefF Attribute (25)	Random Forest	91.91%	0.919	0.094	0.919
ReliefF Attribute (25)	SMO	92.34%	0.923	0.088	0.923
ReliefF Attribute (50)	Bayesian	87.84%	0.878	0.144	0.878
ReliefF Attribute (50)	CART	89.69%	0.897	0.100	0.900
ReliefF Attribute (50)	J48	91.15%	0.911	0.081	0.915
ReliefF Attribute (50)	Random Forest	93.42%	0.934	0.072	0.934
ReliefF Attribute (50)	SMO	93.18%	0.932	0.077	0.932

APPENDIX-C: The Permissions Selected in Different Datasets

Permissions	Dataset-500	Dataset-1000	Dataset-2500	Dataset-3784
android:versionCode	1	1	1	1
android.permission.ACCESS_WIFI_STATE	1	1	1	1
android.permission.BROADCAST_STICKY	1			
android.permission.CALL_PRIVILEGED	1	1	1	1
android.permission.CHANGE_NETWORK_STATE	1	1		
android.permission.CHANGE_WIFI_STATE	1	1	1	1
android.permission.DELETE_CACHE_FILES	1			
android.permission.DELETE_PACKAGES	1		1	1
android.permission.GET_TASKS	1	1	1	1
android.permission.INTERNET	1	1	1	1
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	1	1	1	1
android.permission.READ_EXTERNAL_STORAGE	1	1	1	1
android.permission.READ_LOGS	1	1	1	1
android.permission.READ_PHONE_STATE	1	1	1	1
android.permission.READ_SMS	1	1	1	1
android.permission.RECEIVE_BOOT_COMPLETED	1	1	1	1
android.permission.RECEIVE_SMS	1	1	1	1
android.permission.RECEIVE_USER_PRESENT	1	1	1	1
android.permission.SEND_SMS	1	1	1	1
android.permission.SYSTEM_ALERT_WINDOW	1	1	1	1
android.permission.WAKE_LOCK	1	1	1	1
android.permission.WRITE_APN_SETTINGS	1	1	1	1
android.permission.WRITE_EXTERNAL_STORAGE	1	1	1	1
com.android.browser.permission.WRITE_HISTORY_BOOKMARKS	1	1	1	1
com.android.launcher.permission.INSTALL_SHORTCUT	1	1	1	1
android.permission.INSTALL_PACKAGES		1	1	1
android.permission.RESTART_PACKAGES		1	1	1
android.permission.WRITE_SETTINGS		1	1	1

*The selected permissions are shown with 1 in the table.

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü	□
Sosyal Bilimler Enstitüsü	□
Uygulamalı Matematik Enstitüsü	□
Enformatik Enstitüsü	☒
Deniz Bilimleri Enstitüsü	□

YAZARIN

Soyadı : PEHLİVAN
Adı : UĞUR
Bölümü : BİLİŞİM SİSTEMLERİ

TEZİN ADI (İngilizce) : PERMISSION BASED MALWARE DETECTION IN ANDROID APLLICATIONS

TEZİN TÜRÜ : Yüksek Lisans ☒ Doktora □

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir. □
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden □
   kaynak gösterilmek şartıyla fotokopi alınabilir.
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz. ☒

TEZİN KÜTÜPHANEYE TESLİM TARİHİ : ……………………