# A COMPARISON OF CLASSIFICATION ALGORITHMS FOR MOBILE MALWARE DETECTION: MARKET METADATA AS INPUT SOURCE

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

BY

NURAY BALTACI

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

SEPTEMBER 2014

# A COMPARISON OF CLASSIFICATION ALGORITHMS FOR MOBILE MALWARE DETECTION: MARKET METADATA AS INPUT SOURCE

Submitted by Nuray Baltacı in partial fulfillment of the requirements for the degree of Master of Science in 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

Assist. Prof. Dr. Cengiz Acartürk _____
Co-Supervisor, Cognitive Science, 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:** _____

**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: Nuray Baltacı**

**Signature: _____**

# ABSTRACT

A COMPARISON OF CLASSIFICATION ALGORITHMS FOR MOBILE MALWARE DETECTION: MARKET METADATA AS INPUT SOURCE

Baltacı, Nuray
M.S., Department of Information Systems
Supervisor: Prof. Dr. Nazife Baykal
Co-Supervisor: Assist. Prof. Dr. Cengiz Acartürk

September 2014, 76 pages

The prevalence of mobile devices has been catching the attention of malware authors especially for Android OS supported devices due to its user-centric security policy and open application development strategy for its official application market. In this study, an automated feature-based static analysis method was applied to detect malicious mobile applications on Android devices. The main purpose of the study is to investigate the contribution of other application market metadata to the detection of malicious applications in addition to requested permissions. Hence, the information of applications presented on the official market when a user wants to download them was used as the feature set for training supervised classification algorithms. This feature set includes permissions requested from the user at the installation time, and other metadata about an application including but not limited to application category, download number category, and developer name. Additionally, different classification algorithms were compared in terms of their predictive accuracy and the effect of feature selection algorithms on the improvement of classification task was investigated. Naïve Bayes, k-nearest neighbor, J48 and random forest were chosen as classification algorithms. As filter-based algorithms, Chi-Square, Information Gain and ReliefF feature selection methods were utilized to reduce the number of attributes used to train those classification algorithms.

**Keywords:** Mobile malware detection, Classification, Google market metadata, Machine learning

# ÖZ

KÖTÜ AMAÇLI MOBİL YAZILIMLARIN TESPİTİ İÇİN KULLANILAN SINIFLANDIRMA ALGORİTMALARININ KIYASLANMASI: GİRDİ KAYNAĞI OLARAK MARKET META VERİSİ

Baltacı, Nuray
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Prof. Dr. Nazife Baykal
Ortak Tez Yöneticisi: Yrd. Doç. Dr. Cengiz Acartürk

Eylül 2014, 76 sayfa

Kullanıcı merkezli güvenlik politikası ve resmi uygulama marketi için açık uygulama geliştirme stratejisi nedeniyle Android işletim sistemi destekli cihazlar başta olmak üzere, mobil cihazların yaygınlaşması kötü amaçlı yazılım geliştiricilerin dikkatini çekmektedir. Bu çalışmada, Android cihazlardaki kötü amaçlı mobil uygulamaların tespiti için otomatik, özellik-tabanlı bir statik analiz yöntemi uygulanmıştır. Çalışmanın esas amacı, uygulama tarafından talep edilen izinlerin yanı sıra diğer market bilgilerinin zararlı yazılımların tespitine olan katkısını araştırmaktır. Dolayısıyla, uygulamalar kullanıcılar tarafından resmi marketten indirilirken uygulamalarla ilgili sunulan bilgiler özellik kümesi olarak güdümlü sınıflandırma algoritmalarının eğitilmesinde kullanılmak kullanılmıştır. Bu özellik kümesi, uygulamanın kurulması anında kullanıcıdan talep edilen izinleri ve uygulama kategorisi, indirilme sayısı kategorisi, geliştirici adı vb. gibi uygulamayla ilgili diğer meta verileri kapsamaktadır. Ek olarak, çeşitli sınıflandırma algoritmaları tahminlerinin doğruluğu açısından kıyaslanmış ve özellikle seçme algoritmalarının sınıflandırma görevinin iyileştirilmesi üzerine etkisi araştırılmıştır. Sınıflandırma algoritmaları olarak Naive Bayes, k- nearest neighbor, J48 ve random forest seçilmiştir. Birer filtre-tabanlı algoritma olan Chi-Square, Information Gain ve ReliefF özellik seçme yöntemlerinden ise bahsi geçen sınıflandırma algoritmalarının eğitilmesinde kullanılan niteliklerin sayısını azaltmak üzere faydalanılmıştır.

**Anahtar Kelimeler:** Mobil zararlı yazılım tespiti, Sınıflandırma, Google market meta verisi, Makine öğrenimi

# DEDICATION

I dedicate this thesis to my beloved parents, throughout my life being proud of all my successes and prodding me, to the living memories of my father Kamil Baltacı and always being nearest by me, pandering to my whims and praying for me, my precious mother Neziha Baltacı. I also dedicate my thesis to my sisters Öznur Baltacı Oral and Saliha Baltacı Akgün who have been encouraging and trusting me for all the time, most importantly giving their valuable sisterhood.

# ACKNOWLEDGMENTS

First and foremost I am indebted to my thesis supervisor Prof. Dr. Nazife Baykal for her motivation and support throughout my study. I owe a great debt of gratitude to my thesis co-supervisor Assist. Prof. Dr. Cengiz Acartürk who has guided and encouraged me even from the place he has been commissioned from miles away.

I cannot find the words to express the unconditional love and support of my mother Neziha Baltacı. She has always been with me with her deepest faith and I have always overcome challenges thanks to her. Also it would be understatement if I do not mention that my beloved sisters Öznur Baltacı Oral and Saliha Baltacı Akgün have never let me quit whenever I felt desperate. I am forever grateful to them. I want to thank to my cousin Dilara Avcı for her fellowship and not leaving me alone in Ankara.

This thesis would have remained a dream had it not been for Kamil Akhüseyinoğlu. He helped to develop the Java applications used to collect the data and guided me throughout the whole study. He has always offered his encouragement and supported me spending sleepless nights. I am thankful to him also because of his being always nearby me.

I would like to thank my friend and room-mate Şeyma Küçüközer for her valuable insights, and becoming the driving force behind my thesis. She has always made me believe to complete my study. My colleague Serhat Peker was also another supporter of me by always giving recommendation to study in the library. I want to wish my special thanks to Informatics Institute secretaries Sibel Gülnar and Hakan Güler as they came to my rescue whenever I am in trouble. They all and my other colleagues, Pelin Canbaz, Ali Mert Ertuğrul, Özge Gürbüz, and Emre Sezgin were the source of my exhilaration during my work. As my other friend, my cat Gofret has always cheered me up and relaxed me.

It gives me great pleasure in acknowledging the participation in my defense, valuable feedbacks and suggestions of my examining committee members, Assoc. Prof. Dr. Aysu Betin Can, Assist. Prof. Dr. Aybar Can Acar, Dr. Serkan Alkan and Assist Prof. Dr. Tuğba Taşkaya Temizel. I would also like to show my gratitude to the URAP (University Ranking by Academic Performance) Research Laboratory members, especially to Prof. Dr. Ural Akbulut and Prof. Dr. Canan Çilingir, for their concerns and close interest on my study.

Lastly, I also thank to The Scientific and Technical Research Council of Turkey (TÜBİTAK) for the scholarship during my MSc study.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

**API:** Application Programming Interface

**AV:** Antivirus

**DoS:** Denial of Service

**FNR:** False Negative Rate

**FPR:** False Positive Rate

**FS:** Feature Selection

**IDS:** Intrusion Detection System

**IMEI:** The International Mobile Station Equipment Identity

**iOS:** IPhone operating system

**kNN:** k-nearest neighbor

**Malware:** Malicious Software

**MAP:** Maximum a posteriori

**OS:** Operating System

**PC:** Personal Computer

**ROC:** Receiver Operating Characteristic

**SVM:** Support Vector Machine

**TNR:** True Negative Rate

**TPR:** True Positive Rate

**URI:** Unique resource identifier

# CHAPTER 1

# INTRODUCTION

## 1.1. Background of the Study

Smartphones and tablets have been pervading in both daily use and business settings. Including desktops and laptops, conventional PC market is falling behind the market for mobile computing. Statistics reported by the Intelligence Research Service of Business Insider strongly support this fact that it was totally 28 percent of the world population who owned tablets and smartphones when compared to PC owners constituting the 20 percent at the end of 2013. (Blue Coat Systems, 2014) In spite of their security vulnerabilities, mobile devices do not decrease the speed of pervasion. (Blue Coat Systems, 2014)

The ubiquitous usage of mobile devices has induced the burst of mobile application market because mobile applications enhance the capabilities of mobile devices and improve the customer experience. People take advantage of downloading these applications to their smartphones or other mobile devices for the purpose of amusement, shopping, online banking, business needs, tracking their daily supportive health practices and of almost any purpose one can imagine. This increasing demand for mobile applications is supplied by platforms of official vendors and third party markets. As they produce their official applications, they also realize the importance of the support to meet this demand by third party developers. However, this approach has some security risks because not only the legitimate developers but also malware (malicious software) authors upload applications to these markets.

There are diverse incentives of hackers behind writing mobile malware. Authors may want to damage users in purpose of novelty and amusement. An example of mobile malware, *Ikee.A* changed the wallpaper of iPhones once infected the device and *Smspacem*, an SMS spam, targeted Android devices to send text messages against religion. Other inducements for hacking can be listed as selling user information (location of the user, contacts list, browser history and IMEI), stealing user credentials (like bank account credentials, credit card numbers, and account passwords), SMS spam, search engine optimization, ransom and making premium-rate calls and sending premium rate SMSs. Premium-rate SMSs sent from the phone without user consent can be hidden so craftily that the user may only understand after seeing his/her phone bill. ( Porter Felt, Finifter, Chin, Hanna, & Wagner, 2011)

Mobile application platform providers have different security precautions against the mischievousness of cybercriminals. Google applies an open strategy to let the developers publish their applications whilst Apple strictly reviews the products of their approved developers before releasing. (Bose, Hu, Shin, & Park, 2011) Apple has been effective in their application provenance policy so, actual malware targeting non-jailbroken iOS (IPhone operating system) devices have not seen thus far. (Symantec, 2011) However, Google's user report triggered review process for their application market and user centric permission system makes Google Play (formerly named as Android Market) and Android Operating System (OS) attractive for attackers. In 2012, attacks on Android devices constituted 95% of overall infections and with 32.8 million devices get infected, and doubled this amount compared to the past year. Attackers used application repackaging, SMS phishing and malicious URLs as exploitation methods of OS's commonly. (Rapit7, 2013) Cybercriminals will continue their attacks in an evolving manner targeting Android OS owing to the unregulated application market structure and diversity of Android based devices. (Blue Coat Sytems, 2013)

Mobile application platform providers are capable of removing malicious applications from the market quickly when detected and they have the right of remotely uninstalling these applications from the downloaded devices. Despite quick actions of them, the time window between the placement and detection of malicious application may cause unwanted damages to a lot of users. (Blue Coat Systems, 2014) For instance, attacks named as "*Android.Rootcager*, *Android.Pjapps* and *Android.Bgserv*" during 2010 and 2011, targeted legitimate applications by injecting malicious code into these applications and replacing the digital signatures with an uncertified ones. These trojanized applications which were put on the official Android Market or third-party markets harmed hundreds of thousands of users. (Symantec, 2011) Another threat on Google's official market was discovered in early 2010 as malicious applications developed by the developer named Droid09. These applications deceived users by masquerading themselves as legitimate mobile banking applications and then stole banking credentials of users by using phishing techniques. (Sybase, 2011) After realizing, Google quickly removed 51 applications belonging to the same developer. (Symantec, 2012)

## 1.2. Purpose of the Study and Research Questions

In this study, the performance of different machine learning classification algorithms on Google metadata-based static analysis method is evaluated. Mainly, the contribution of Google Play Market specific application information to the effectiveness of Android permission-based detection model is questioned. Since Google metadata on Google Play market includes information about permissions required when downloading applications and other information like developer name, download number of application, user rates, minimum required Android API level and so on, the contribution of other Google data mentioned here is investigated by dividing dataset according to permission features and others. The performance comparison is made on these two dataset. Also, the effects of feature selection methods with filter approach on the performance of classification algorithms are observed. The following research questions are attempted to be answered throughout the study:

1) Is it possible to accept Google Play market metadata as meaningful attribute for a supervised machine learning algorithm used for mobile malware detection?
2) Which classification algorithm has the highest accuracy for the Android malware detection problem by using Google Play market metadata among Naïve Bayes, k nearest neighbor, random forest and J48?

3) Which classification algorithm, feature selection method and the number of selected features combination is most accurate in static detection of Android malware with Google metadata? Chi-Square, ReliefF and Information Gain Score are the feature selection methods used with classification algorithms.

## 1.3. Significance of the Study

Proliferation of mobile devices has caused the exploitation of them by cybercriminals to obtain immense amount of profit by confidential personal and financial information. People have been deserting PCs and increasingly preferring mobile devices as the means of accessing data. While using mobile devices, users utilize from mobile applications in order to perform tasks like sending SMS, surfing on the internet, making online banking transactions, taking pictures and so on. This makes mobile devices and applications prone to mobile malware because they are full of personal and financial data to gain access by cybercriminals. (Rapit7, 2013) Even in recent years, malware industry has shown inconceivable progression so that it turned into a highly illegal economy sustaining under supply and demand laws like traditional market-based economies. In this market, cybercriminals sell the data they steal, rent botnets, sell newly explored vulnerabilities and exploit kits. (Blue Coat Systems, 2014)

Hackers have a wide range of subtle methods to harm mobile devices and trick users. They can exploit the vulnerabilities of mobile operating systems like the ones enabling buffer overflows by sending more data than the memory can handle and causing the propagation of malware to other areas in the phone. ( Lawton, 2008) Another example of hackers' craftiness is that they can modify legitimate applications to include hidden channel and then leak information from the victim's phone. This threat is called as man-in-the-middle attack and can be achieved by the installation of malicious certificates and reconfiguring proxy settings and performing other modifications. (HP, 2013) These are only some well-known examples to the weapons of cybercriminals and they add new ones into their ammunition every day. As counterattack to this growing army, security solutions proposed by researchers have been increasing. However this research field is immature and needs to be explored deeply. (La Polla, Martinelli, & Sgandurra, 2013)

The fast growth of mobile market, transformation of mobile malware into huge market economy and the low saturation of the mobile malware research field have made the way for realizing a study in mobile malware detection. In addition, the difference of official application market policies applied by mobile application platform and mobile OS providers have an effect on the proposed model in this study. For example, Apple Incorporation applies manual inspection of mobile applications by security experts before presenting them on the Apple Store. (Abu Samra, Yim, & Ghanem, 2013) On the other hand, Google has more passive security policy allowing anyone to publish application on their official market (Abu Samra, Yim, & Ghanem, 2013) despite their user reporting mechanism for suspicious applications and Bouncer for automatically scanning applications prior to upload. (Petsas, Voyatzis, Athanasopoulos, Polychronakis, & Ioannidis, 2014) This arises the question whether the applications' metadata presented on the official application market can be used as the indication of malicious content or not.

Additionally, when a user downloads an Android application to his phone, a list of permissions is presented by that application in non-technical language at installation time. Since Android has no middle way of granting permissions to applications, users have to grant access to all the

requested permissions if he wants to install the applications. Anroid's permission system passes security risks on users and gives freedom of choice to decide on the safety of applications' permissions. (Symantec, 2011) However, users are generally not competence enough to make decision about permissions, so guidance is needed before installing applications. This user centric approach may become challenging and users may be stimulated to accept every permission requested by an application causing security risks. (Bose, Hu, Shin, & Park, 2011)

In the lights of the things mentioned above, the need for a static feature based mobile malware detection system for Android devices considering requested permissions and Google Market data, including developer name, download time, user ratings, and so on, became an incentive behind this study.

## 1.4. Definition of Terms

**Hash value:** Hash value, consisting of numerical and/or alphabetical values and fixed in size, is used to ensure the integrity of files (or message) and so to determine whether a file has been tampered while being sent through insecure channels. (Microsoft, 2014)

**IMEI:** A unique number used to identify phones.

**Jailbroken device:** For devices having Android OS, users can root devices by circumventing built-in limitations related to security and OS use of the devices. (USA Department of Commerce, 2012)

**Malware:** Malware, short for malicious or malevolent software, is software used or programmed by attackers to disrupt computer operations, gather sensitive information, or gain access to private computer systems.

**Premium-rate phone calls and SMS:** For the definition of this term, the following definition was adopted from the study of ( Porter Felt, Finifter, Chin, Hanna, & Wagner, 2011) (p. 6) "Legitimate premium-rate phone calls and SMS messages deliver valuable content, such as stock quotes, technical support, or adult services. The cost of a premium-rate call or SMS is charged to the sender's phone bill. Premium-rate calls can cost several dollars per minute, and premium-rate SMS messages can cost several dollars per message."

**Search Engine Optimization:** For the definition of this term, the following definition was adopted from the study of ( Porter Felt, Finifter, Chin, Hanna, & Wagner, 2011) (p. 7) "Malware can be employed to improve a web site's ranking in search engine results. This type of malware sends web requests to the search engine for the target search term. The malware then fraudulently "clicks" on the search result that corresponds to the target web site. As a result, the web site's rank for that search term will increase. The value of fraudulent search engine optimization depends on how well the target site can capitalize on its increased visibility, but search engine optimization is a large and lucrative market."

**Ransom:** For the definition of this term, the following definition was adopted from the study of ( Porter Felt, Finifter, Chin, Hanna, & Wagner, 2011) (7) "Mobile malware that seriously threatens or publicly embarrasses the user for profit"

# CHAPTER 2

# LITERATURE REVIEW

This chapter presents literature review and starts with the security policies and practices of the two widely used mobile device vendors and application platform providers. Following this, Android OS, application structure and permissions system are explained to shed light on the ground of the studies in this area. Finally, malware detection methods and the studies about them are presented.

## 2.1. Application Market and Operating System Security Policies and Practices

This section explains and compares the security policies and practices followed by the two pioneer vendors of mobile devices and their official application markets, namely Apple-AppleStore and Android-GooglePlay.

A software developer is required to register and pay annual licensing fee and then obtain a digital certificate of Apple to be able to release software for Apple products. On the other hand anyone can publish application on the official Android Market with Google's passive publication mechanism. ( Wu , Mao, Wei, Lee, & Wu, 2012) Like iOS, Android OS requires digital signatures to install and run applications but the certificates are not Google-issued and the developers can generate digital signatures as often as they want, give any company name and contact information in the certificates they like and this makes the traceability of hackers virtually impossible. Google's signature mechanism causes two problems. Since traceability of applications becomes harder, it gets easier to generate and distribute malware. In addition, by adding malicious code into an existing legitimate application and signing it with the anonymous certificate makes the addition of Trojan horses into benign applications an easy job for attackers. (Symantec, 2011)

Apple reviews every application before publication and applies code signing model to prevent attackers from modifying or infecting benign applications. (Symantec, 2011) Google also checks applications simply but the process is not as strict as Apple's process and adopts a strategy of deleting applications from the market after they are found to be malicious. (Bose, Hu, Shin, & Park, 2011) ( Porter Felt, Finifter, Chin, Hanna, & Wagner, 2011) ( Wu , Mao, Wei, Lee, & Wu, 2012) In such a strategy, hackers find the required time gap to download and to alter the legitimate application into a malware by injecting malicious code. This hijacking

method makes the ways simple for hackers and it gets even more difficult to detect malwares. (Symantec, 2012)

In addition, Apple applies a kill switch to make malicious applications inactive because illegitimate applications may infect Apple devices despite their strict policy. Kill switch is a precaution taken to remotely deactivate or remove applications from the mobile devices by application platform providers in case of a malicious content is realized to guard mobile device users against security threats. Android uses a similar method with Apple's kill switch to remove application from Android devices remotely. (Bose, Hu, Shin, & Park, 2011)

Lastly, Android allows installing third-party applications that may increase the spread of Android malware. ( Wu , Mao, Wei, Lee, & Wu, 2012)

To sum up, there are some differences and similarities of security policies among the two pioneer application platform (and OS) providers, Apple (iOS) and Google (Android). While Apple applies digital certification with annual fee for developers, Google's certification is more flexible and causes the propagation of malware easily. In addition, Apple manually and strictly inspects the applications' code before uploading them on the market, but Google applies more open reviewing strategy and removes the illegitimate applications from the market afterwards. Whereas Android allows the download of applications from third-party market, iOS restricts the users from this aspect. Both Google and Apple have kill switch mechanism to remove malicious applications remotely from mobile devices.

## 2.2. Android Operating System

Running top of Linux kernel which is an open source Unix-like operating system, (wikipedia, 2014) Android is a middleware and an operating system for mobile phones. ( Enck, Ongtang, & McDani, 2009) In Android OS, applications are strongly isolated from the system and each other by the customization of underlying Linux internals, and this mechanism is called sandboxing. ( Enck, Ongtang, & McDani, 2009) (Bose, Hu, Shin, & Park, 2011) Mobile applications for Android are written in Java language and execute on the OS with unique user and group identity (UID) assigned at installation time on their own Linux processes. ( Orthacker, et al., 2011) (Bose, Hu, Shin, & Park, 2011) ( Enck, Ongtang, & McDani, 2009) Assignment of UID to applications assures the sandboxing mechanism which restricts the access to the file system resources and memory. Applying a fine-grained permission system, Android compels restrictions on communication, share of resources and functions. If the user provides access to the required permissions demanded by applications, then applications can access to resources. Additionally, by the help of this isolation mechanism, the effects of buffer overflows are minimized. In other words, if an application is infected due to the exploitation of vulnerability in OS, other parts of the system and applications are protected. ( Enck, Ongtang, & McDani, 2009)

## 2.3. Android Application Structure

Android applications are packaged files which are called as apk files, and brought in zipped form. The assets and resources in the form of multimedia to be used for the user interfaces and functions, the Dalvik executable (dex file) containing byte code and the configuration file, Android Manifest file, are contained in an apk. ( Schreckling, Huber, Höhne, & Posegga, 2013)

There exist four types of basic components to be used to build an application which are *activities, services, broadcast receivers* and *content providers*. User interfaces are formed by a*ctivities* and only one activity can be active at a time if an application contains many activities. *Services* are in job to fulfill background or time consuming tasks and are interacted by API functions by triggering remote calls. In order to share their data with other applications, applications use *content providers* and to retrieve this data, they use content resolvers. Content providers query data of another content provider by using URIs (unique resource identifier). *Broadcast receivers* provide the exchange of intent messages, intent to perform an action, between applications. ( Schreckling, Huber, Höhne, & Posegga, 2013) These four types of components use intent messages to communicate and their communication mechanism is called as IPC (inter-process communication). ( Enck, Ongtang, & McDani, 2009)

## 2.4. Android Permissions System

As it is stated in the Android OS part, applications face the restriction on system resources, functions and communication. To be able to access these restricted elements, applications must declare permissions in their manifest files. (Abu Samra, Yim, & Ghanem, 2013) (Bose, Hu, Shin, & Park, 2011) Several hardware devices like GPS and camera, sensitive parts of the OS like contacts and the parts of other applications to be accessed can be thought as the elements which have restrictions. For instance, to access the internet, "android.permission.INTERNET" statement should be placed in the manifest of an apk while the permission "android. Permission.READ CONTACTS" is used by an application to access the contacts of a user. (Abu Samra, Yim, & Ghanem, 2013) The user, who wants to install an application, should grant all of the permissions requested by that application at the installation time, not at the runtime, and there is no way to grant the some part of the permissions. ( Meurer & Wismüller, 2012) Those granted permissions are enforced when the application executes and the permissions which are not granted yield errors. (Bose, Hu, Shin, & Park, 2011)

Android permissions are grouped into three categories according to their security risk levels as normal, dangerous and signature/system. ( Schreckling, Huber, Höhne, & Posegga, 2013) ( Zhu & Peiravian, 2013) ( Meurer & Wismüller, 2012) Normal permissions do not threaten the security of mobile devices and hence they are not asked to the user to be approved, in fact granted without notification. On the other hand, dangerous and system type permissions are approved by the user at install time because they have the control on the restriction of critical resources and private data. Signature or system permissions can be requested by only the applications pre-installed and signed by the device manufacturer and not accessible for normal developers because they control the access to the main system services and data. ( Schreckling, Huber, Höhne, & Posegga, 2013) ( Meurer & Wismüller, 2012) Beside these predefined permissions for Android applications, developers may also define their own permissions like in the case of having purpose to protect a content provider. ( Meurer & Wismüller, 2012) (Bose, Hu, Shin, & Park, 2011)

## 2.5. Malware Detection Methods

Static and dynamic analyses are the two main types of malware detection techniques that analyzes both PC and smartphone malware according to the way of code analysis. (Suarez-Tangil , Tapiador, Peris-Lopez, & Blasco, 2014) ( Wu , Mao, Wei, Lee, & Wu, 2012) Static analysis is a detection method which comprise of unpacking and disassembling or decompiling the malware samples and inspecting the obtained code. On the other hand, dynamic analysis

handles specimens by running them in a controlled environment and tries to find out malicious behaviors. Dynamic and static analyses are conducted by extracting and analyzing a number of features as a result of sample inspection. From this point of view, by using several features as attributes, machine learning and data mining approaches have been introduced as automated malware analysis techniques to assist analysts in carrying out classification and clustering tasks. (Suarez-Tangil , Tapiador, Peris-Lopez, & Blasco, 2014)

Wu , Mao, Wei, Lee, and Wu (2012) discuss malware detection as two different types as misuse detection and anomaly detection. Misuse detection is the method of applying rules or policies based on matching the signatures of malware with the ones in database. It precisely detects the Android malware in case of signature match but needs to update the signatures. Anomaly detection is defined as applying machine learning algorithms to learn behaviors of known malware and to predict unknown malware but it sometimes causes high false positive. ( Wu , Mao, Wei, Lee, & Wu, 2012)

### 2.5.1. Dynamic (Behavior-Based) Analysis

A behavioral framework for detecting mobile viruses, worms and Trojans targeting Symbian OS is proposed by (Bose, Hu, Shin, & Park, 2008). A database of behavioral signatures is constructed by collecting system events and resource-access attempts made by the program and by applying *temporal logic of causal knowledge* (TLCK) method on those key behavior signatures of malwares reported to date. The interested behaviors are the ones presenting malicious activity like draining the battery, overwriting system files, installing a worm payload, sending infected messages etc. and these behaviors are not enough to label an application as malicious in isolation. Hence, the logical ordering of these activities in time is under inspection by using TLCK method. The monitoring layer is implemented on a Symbian emulator to collect run-time behavior signatures. A classifier, SVM (support vector machine), is trained from normal and malicious applications and evaluated on both emulated and real-world malicious data. Their results show high detection rates with novel malwares which have certainly matching behavioral patterns with the ones in the database. (Bose, Hu, Shin, & Park, 2008)

Another behavioral framework, *Andromaly* is a host-based and lightweight (in terms of CPU, memory and battery consumption) malware detection system designed to monitor and obtain events and features from the mobile device, and then to perform real-time anomaly detection. Andromaly uses system metrics such as CPU consumption, number of sent packets through the Wi-Fi, number of running processes, battery level etc. as behavioral features. Chi-Square, Fisher Score and Information Gain methods were applied to select features and as machine learning algorithms k-means, logistic regression, histograms, decision tree, Bayesian networks and Naïve Bayes were chosen. At the time of their analysis, there was not known Android malware sample, so they developed four malicious applications that perform DoS and information theft attacks. They evaluated the different combinations of feature selection methods, the predefined feature number and machine learning algorithms and compared their performances. According to the empirical results, Andromaly is effective for mobile malware detection and the best configuration is the Naïve Bayes trained with top 10 features selected by the Fisher Score algorithm. ( Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012)

Behavioral-based detection systems may cause extra cost in deployment environment, because mobile handsets have limitation in terms of CPU and battery. To overcome this obstacle, (Damopoulos, Kambourakis, & Portokalidis, 2014) presents a proof-of-concept mobile IDS

(intrusion detection system) architecture deployed both on the host and the cloud. Their architecture consists of event sensors which collects events from the device to construct behavioral signatures [like system calls, inter-process communications, hardware sensors, API calls, system services (e.g. user SMS), and any library call], system managers to collect signatures and forward to decision engines, decision engines to fulfill detection mechanism on device, cloud manager to apply decision algorithms on the cloud. Their system decides based on sensitivity, which changes according to the user preferences on system sources to provide flexibility. Four different detection mechanisms from the previous work, namely SMS Profiler, iDMA, iTL and Touchstroke are used and then Random Forest algorithm is used as the classification algorithm. They use the Performance (CPU and memory consumption) and Timeliness (train and test time) metrics to evaluate the effectiveness of their real-time IDS. Cloud based detection is found to have lower training and testing time but overall detection time of cloud is worse than host-based detection because of communication delays. However CPU performance shows the battery lifetime is affected by on-device detection. As a result, they conclude that a hybrid solution performing the most heavyweight detection tasks on the cloud and the more time sensitive ones on the host would be a better solution. (Damopoulos, Kambourakis, & Portokalidis, 2014)

### 2.5.2. Static Analysis

URANOS, an application rewriting framework developed for Android devices by ( Schreckling, Huber, Höhne, & Posegga, 2013) lets the users of mobile phones to selectively deactivate the permissions of applications according to their preferences without rooting or manipulating their smartphones. This framework analyzes the byte code of the application in order to infer the required permissions during execution and compares with the permissions requested in the application manifest file. As a result of this on-device static analysis, excessive permissions are detected and shared with the user. According to the decision of the user whether to enable or disable permissions, URANOS rewrites the application byte code. This study expresses the importance of guidance by displaying excessive permissions to the users for helping them make informed decisions because granting excessive permissions to applications gives the way for new exploit techniques. ( Schreckling, Huber, Höhne, & Posegga, 2013) As another guide to the smartphone users, *Kirin* security service for devices with Android OS developed by ( Enck, Ongtang, & McDani, 2009) uses a set of predefined security rules and performs lightweight certification at install time. These rules are based on security configuration available in an application's manifest file (permissions, intents, and application components) and defined clearly by specifying unwanted configurations and their combinations. Thus they followed security requirements engineering processes (identifying assets, functional requirements, determining assets' security goals and threats, developing assets' security requirements, determining security mechanism limitations and adjusting security rules accordingly) to specify rules and then proposed a security language to define their semantics. Results show that certification technique fails for only 1.6% of applications in their dataset hence Kirin can be reasonable for practically mitigate malware. ( Enck, Ongtang, & McDani, 2009)

An Android application named as APEFS was also developed to guide the users to make decision while downloading applications from Google's official market. This application does not require the root access to the phone and downloads applications via Google Play. In order to do so, APEFS parses needed information from the details page of the application like developer name, price, rating and the requested permissions. Users can define profiles in the APEFS according to their need of security and privacy and for nonprofessional users it includes pre-

defined permission profiles. These profiles are used to filter the applications when a user searches the market and to find the ones which do not match the security level of the user among the already installed applications. Then user can delete those unsuitable applications from the phone. ( Meurer & Wismüller, 2012)

DroidMat, developed by ( Wu , Mao, Wei, Lee, & Wu, 2012) is a system used as a static feature-based mechanism to detect Android malware by considering requested permissions, intent messages passing, API calls and components of applications (activity, service and receiver) as static information. The dataset used in the study includes 238 Android malware collected from a public Android dataset, Contagio mobile, and 1500 benign applications downloaded from official Android market and verified through the website of VirusTotal malware detection community . As the first step, K-means and EM clustering methods are used to enhance the malware modeling capability. The number of clusters is determined by using Singular Value Decomposition method. Then kNN (k-nearest neighbor) and Naïve Bayes algorithms are trained and tested to detect unknown malware samples. Different combinations of static features are tried and the feature set consisting of permissions, intent messages and API calls is found to be the most precise. Finally, the combination of k-means as the clustering and kNN with k=1 as classification algorithm is chosen as best result. ( Wu , Mao, Wei, Lee, & Wu, 2012)

In order to improve the prediction accuracy of the permission-based detection method, a framework having the combination of requested permissions and static API call behaviors as the feature set for machine learning classification tasks was proposed. ( Zhu & Peiravian, 2013) They utilized the validated dataset of a former study, Malware Genome Project, to make their analysis and built three benchmark datasets (one containing only permissions as feature set, the other one containing only API calls and the last one containing combination of them) to evaluate the contribution of static API calls. By extracting permissions from AndroidManifest file and API calls from class files, they used them as the attributes to train different classification methods which are SVM (support vector machine), decision tree and bagging algorithms. Bagging algorithm has the best performance for all of benchmark datasets and they ground this performance to the capability of bagging algorithm's on imbalanced dataset in terms of class attribute. (Their dataset includes more benign applications than malicious applications) ( Zhu & Peiravian, 2013)

Those works mentioned above and the others which are not handled here are acceptable and effective enough to explain the malicious characteristics of Android applications by using combinations of Android permissions, native code, embedded applications, and application components. (Glodek & Harang, 2013) extended those works by adding the frequent combinations of such static features as training attributes for random forest classification algorithm. They chose random forest because of its high accuracy performance and computational efficiency. They also used the dataset of Malware Genome Project for labeled malwares and collected legitimate applications from third party markets randomly. They came to conclusion that the combinations of permissions seen frequently in the dataset improve previous results with true positive rates above 90% and with acceptable false positive rates. (Glodek & Harang, 2013)

In one of the studies using static analysis technique, researchers apply a new method called *Activation Patterns* by extracting the permissions and other metadata (like description of application, download count, price and category of each application) of 130.211 applications

collected from Google's official application market. They deeply inspect the permissions and their relations by executing semantic search queries to figure out anomalies and identify the clusters of similar applications according to permissions and descriptions of them with Growing Neural Gas clustering algorithm. By using activation patterns method they try to find out anomalies utilizing generated patterns. For example they look into the relationship between the description of and permissions requested by an application. As a result, their study constructed a solid basis for further anomaly detection of applications on the market and their clustering based on permissions were promising because of giving information about the typical usage of permissions by various application categories. (Bose, Hu, Shin, & Park, 2011)

In their research, (Abu Samra, Yim, & Ghanem, 2013) use a dataset which includes approximately 188,000 applications downloaded from Google's Android Market in November 2011 by using web crawling technique for a former study. This dataset had been comprised of top free and top paid applications and then more applications obtained as a result of search done by using some random search terms in the Android Market were added to this dataset. The authors choose to study with applications under business and tools category and justify this selection as they interested in clustering data to two clusters as malicious and non-malicious applications. They use features extracted from xml files of decompressed apk.s and Android Market specific features (like app. Name, category, description, rating values, price etc.) to apply an unsupervised k-means clustering algorithm. The result of k-means algorithm obtained in this article is claimed to give good performance for clustering in Android applications to detect malicious applications.

## 2.6. Summary

The previous work on mobile malware detection is concentrated on two types of methods as dynamic and static analysis. Static analysis technique is the reverse engineering of samples which decompresses applications and inspects the static features of obtained application parts like byte code and permissions requested from users. Signature-based analysis method is the subtype of static analysis which compares the unknown samples with the known malware database and matches the signatures. In this study, a feature-based automated static analysis method is used to detect Android malware because of the low resistance of signature-based method to polymorphic and unseen malware. ( Zhu & Peiravian, 2013) Another main type of detection method, dynamic analysis, examines the applications' behaviors on run time by executing them. Since it requires complicated skills and manual investigation ( Wu , Mao, Wei, Lee, & Wu, 2012), it is not preferred in this study. The reason for choosing Android devices and its official market to inspect lies behind the security policies of them. These devices expose more threats because of the official market's open strategy in terms of application reviews and user-centric permission grant mechanism for applications installed on them. (Bose, Hu, Shin, & Park, 2011) ( Porter Felt, Finifter, Chin, Hanna, & Wagner, 2011) ( Wu , Mao, Wei, Lee, & Wu, 2012) (Symantec, 2011)

Several features have been used to conduct static analysis for mobile malware detection on Android devices in prior studies. Subsets of the combinations for permissions requested from users at install time, intent messages of applications to trigger events, main components of Android applications (content providers, broadcast receivers, activities, and services), and API calls obtained as the result of byte code inspection were used by ( Enck, Ongtang, & McDani, 2009) ( Wu , Mao, Wei, Lee, & Wu, 2012) ( Zhu & Peiravian, 2013) as inputs for their models. (Glodek & Harang, 2013) proposed the use of combinations for permissions seen frequently in

the dataset. Also in this study requested permissions are used as static features as the previous studies have proved their contribution in Android malware detection task. The other static features used by those studies could not be included in this study because all of the applications collected from the official market did not let to be decompressed. However, official application market metadata have been proposed in this study as additional features to requested permissions because the permissions are not sufficient to explain malicious behaviors on their own. ( Meurer & Wismüller, 2012) ( Enck, Ongtang, & McDani, 2009) There exist other studies using official market metadata as inputs for statically detecting Android malware like the ones performed by (Abu Samra, Yim, & Ghanem, 2013) and (Bose, Hu, Shin, & Park, 2011). However they applied unsupervised machine learning techniques, while in this study supervised classification algorithms are applied with and without feature selection algorithms and then their performances are compared.

# CHAPTER 3

# RESEARCH METHODOLOGY

This chapter explains the research methodology and the process to obtain the final baseline datasets. It starts with the explanatory information about detection methods, classification and feature selection algorithms and the justifications for the chosen method and algorithms. Then the following sections clarifies how the data were collected and processed to be used in algorithms, parameters for evaluating the performance of the models and concludes with a pilot study conducted to obtain the baseline datasets for further analyses.

## 3.1. Detection Method

As it is stated in the literature review part, the two basic methods when investigating the malicious behaviors of software or mobile applications are the static and dynamic analysis (detection) methods. Dynamic detection method is applied by running an application on an isolated environment and observing behaviors of the application in order to discover the matching behavior profiles of applications with known malware. (Bose, Hu, Shin, & Park, 2008) Despite the fact that dynamic (behavioral) detection method is more resilient to polymorphic malware owing to the share of similar behavioral profiles of malware variants in the same family, (Bose, Hu, Shin, & Park, 2008) it requires complicated skills and so is costly and time consuming. ( Zhu & Peiravian, 2013)

The chosen detection method in this study is static analysis because of the fact that whereas dynamic analysis can present better understanding of what is going on, but with high cost of deployment environment and manual probing, static analysis reduces the cost and improves the performance. ( Wu , Mao, Wei, Lee, & Wu, 2012) The proposed method is a kind of automated feature based static analysis by regarding permissions and other Google metadata information of Android applications. Under the static analysis, there exists a signature based detection method which has been implemented for many years by researchers and antivirus companies. Signature based detection method relies on the comparison of applications against a list of known malware and has low resistance on unseen, polymorphic, obfuscated and metamorphic malware. ( Zhu & Peiravian, 2013) It is usually proper for post-infection cleanup ( Zhu & Peiravian, 2013), so was not preferred in this study. Another drawback of signature based detection method is being inefficient in terms of battery power consumption which is a scarce resource on mobile devices because of requiring the comparison of each derived signature with the ones in

the database. (Bose, Hu, Shin, & Park, 2008) Moreover, by using the features extracted from the applications, machine learning methods permit the automated detection of applications and have been found to be more accurate than the signature-based approach. (Shabtai, Fledel, & Elovici, 2010) (Suarez-Tangil , Tapiador, Peris-Lopez, & Blasco, 2014) Hence, instead of using the signature-based static detection method, a feature-based static analysis technique is preferred in the present study.

To apply machine learning algorithms, Google metadata of Android applications on Google Play market including the requested permissions by applications at install time is chosen as feature set. Beside permissions, other metadata of applications were chosen because permissions are not sufficient to explain malicious behavior of applications on their own. ( Enck, Ongtang, & McDani, 2009) ( Meurer & Wismüller, 2012) A plausible explanation for this is given by ( Orthacker, et al., 2011). They claim that an application may capable of reaching the system resources which are not the permissions requested for at installation, by utilizing inter-process communication. Hence the applications have actually more capability than implied by their requested permissions owing to the spreading of permissions among them.

## 3.2. Classification and Feature Selection Algorithms

In this section, the supervised classification algorithms and feature selection methods chosen to answer the research questions are explained briefly and their advantages and disadvantages are given. They are selected to work with motivated by their advantages mentioned here and the former studies applied them. Also the implementation specific issues for classification and feature selection algorithms are handled here.

### 3.2.1. Classification Algorithms

#### 3.2.1.1. Naïve Bayes

In machine learning problems, an optimum hypothesis is aimed to be found among candidate hypotheses space H. Here the term optimum corresponds to have maximum probability for a hypothesis. Hypotheses question which class a test instance belongs to for classification tasks. For Bayesian learning algorithms, posterior probabilities of candidate hypothesis are calculated by using the below formula (Bayes theorem) and then a MAP (maximum a posteriori) hypothesis is selected among them. (Mitchell T. M., 1997)

$$p(h|\mathrm{D}) = \frac{p(\mathrm{D}|h)p(h)}{p(D)}$$

Here $p(h|\mathrm{D})$ represents the posterior probability of a hypothesis given the training data D. For this study 2 hypotheses are tested; whether an application belongs to malicious class or benign class. $p(h)$ denotes the prior probability of the hypothesis without any data provided. Any value based on background information can be assigned to this prior probability. If there is no such knowledge, then each class can be assumed to have equal probability for the prediction of a test instance. $p(\mathrm{D}|h)$ is the probability of observed data given the hypothesis. In other words, it represents the multiplication of probabilities for attribute values a test sample have given the class value. Multiplication can be done owing to the independence assumption of each attributes. $p(D)$ is the probability of training data, without in conjunction with any class value. When it comes to select MAP hypothesis, $p(D)$ is left out because it does not have dependency on candidate hypothesis $h$. (Mitchell T. M., 1997)

The Naïve Bayes classifier makes an unrealistic assumption that each attribute (or feature, or predictor) is independent from each other and is equally important on the decision of class value. (Witten, Frank, & Hall, 2011) This can be seen as a drawback at the first glance. However this assumption does not decrease the accuracy of the algorithm seriously, Bayesian learning method works well in practice (Witten, Frank, & Hall, 2011) and it actually outperforms more complicated algorithms. (Shmueli, Patel, & Bruce, 2010) In addition, it is an easy, fast and computationally efficient classification algorithm (Shmueli, Patel, & Bruce, 2010) having a complexity value which is linearly proportional to the number of instances and attributes. (Elkan, 1997)

However one problem exists about Naïve Bayes. If the one of the attribute categories does not exist in the training set, a test sample having that category value will have the zero probability and this attribute will "out vote" the other ones. (Shmueli, Patel, & Bruce, 2010) Out voting of this probability to others can be explained by looking to the way $p(\mathrm{D}|h)$ is calculated. As it is explained above, $p(\mathrm{D}|h)$ is denoted as the multiplication of the each attribute value's probability based on the independence assumption of attributes. If one of the attributes has not occurred in the training set, its probability will be zero and cause $p(\mathrm{D}|h)$ to be zero. To solve this problem, *Laplace estimator* is used. The logic behind Laplace estimator is to add a constant (simply 1) to each numerator of probabilities, and then the total amount which has been added to all numerators is added on to the denominators of probabilities. (Witten, Frank, & Hall, 2011) To clarify this, an example situation can be thought for this study. If we assume that in the training set, there was no malicious instance having the "Access bookmarks" permission, then its attribute count (numerator of conditional probability) would be zero. Hence, a test instance having the "Access bookmarks" permission would have the zero probability with respect to malicious class value no matter what other attribute probabilities are. To handle this problem, count of each attribute value (also the count of other permissions and official market metadata) can be increased by 1, and then the number of considered class value (in this case the number of malicious applications), as being the denominator, can be increased by total number of attributes (because each numerator for each attribute is increased by one). This approach has a disadvantage that there is complication about the amount of constant value added and so about the assignment of prior probabilities. However, if the training samples are admissible in amount, the prior probabilities do not make much difference in practice. In addition, people prefer to add 1 to all initial counts for Laplace estimator. (Witten, Frank, & Hall, 2011)

Naïve Bayes classifier specific implementation issue is to discretize numerical attributes; otherwise the algorithm does not run. Discretization process is explained in detail in the data preprocess part.

#### 3.2.1.2. k-Nearest Neighbor

kNN is an instance based learning algorithm and instance based learning algorithms do not explicitly apply a target function on training data. The only task in terms of training for them is to hold training instances. When a new test instance is found, then the learning algorithm searches the stored training samples to label this test instance considering the target function. They are also called as "lazy learning" algorithm because of postponing the learning task until a new instance is encountered. The main idea of kNN is to calculate the distance of the queried sample to the training instances by using a distance function to find the identified number of nearest neighbors. Then the sample is assigned to a class according to the majority of classes which its nearest neighbors belong to. (Mitchell T. M., 1997)

The most conventional drawback of kNN is its high cost in computation time for making predictions on testing data because it computes the distances of all testing instances to all training instances. (Bhatia & Vandana, 2010) (Witten, Frank, & Hall, 2011) So, in this study first adjustments on the dataset, to form final baseline datasets, were done by using computationally efficient Naïve Bayes algorithm instead of kNN. Another disadvantage of kNN algorithm is that it calculates the distance of a test instance considering all of the attributes in the dataset. In this approach, the most relevant attributes to classify instances are dominated by other useless features. This problem is also called as *curse of dimensionality*. (Mitchell T. M., 1997) Therefore, the data points belonging to the same class in reality can be assigned to different classes as the result of misleading similarity measure. There are some counter measures to eliminate this problem. (Mitchell T. M., 1997) As the first option, the attributes can be given different weights according to their importance for the calculation of distances. Those weights are adjusted by using cross-validation method, leaving a proportion of training set to decide weights and changing iteratively this subset to test the results, in a way that the classification error will be minimized. The second option is to remove irrelevant features completely. This second method corresponds to set the weights of irrelevant features to zero in other approach. (Mitchell T. M., 1997) In this study, feature selection methods are preferred, not only to deal with this kNN-specific problem but also to prevent misleading of irrelevant features for other classifiers and decrease the computational complexity.

Besides its disadvantages, kNN is advantageous from some aspects. First of all, it can be considered as a simple, easy to learn and effective learning algorithm. (Bhatia & Vandana, 2010) (Witten, Frank, & Hall, 2011) Additionally, it can be adapted to wide range of practical problems. ( Dini , Martinelli, Saracino, & Sgandurra, 2012) Another advantage of kNN is that new instances can be added to the training set anytime (Witten, Frank, & Hall, 2011) owing to the principles of lazy learning. Addition of extra training sample afterwards would not affect the learning task because learning occurs as being triggered by test samples. This delayed learning approach also lets the estimation of target function based on each new sample in a local way rather than generalizing them only once to the whole dataset. (Mitchell T. M., 1997)

To implement kNN classification algorithm, one should consider the selection of distance function and *k* value (number of nearest neighbors). There exist different distance functions like Euclidean distance, Manhattan (city-block) distance, Chebyshev distance for numeric attributes and Jackard, Hamming distance for categorical attributes. As it is stated at data preprocessing part, the arranged feature set completely consists of categorical variables, and then a distance function for categorical variables was needed. However in Weka, distance measures are limited, there are not all the functions one may want to use for calculations. In spite of being used for numeric values, Euclidean distance in Weka is adopted for categorical variables. It calculates the distance between data points with Euclidean formula by accepting the distance between two different categorical variables as 1 and the distance between the two same categories as 0. (Weka, 2009) Hence, Euclidean distance was chosen in this study to train k Nearest Neighbor. In addition, selection of the *k* parameter was fulfilled by Weka. It chooses the best smallest k among all the possible k values (from 1 to number of instances) giving minimum error rate. Additionally, before running kNN algorithm, numerical attributes should be normalized to prevent the dominance of attributes with high scaled values on the other ones. However, all the numerical attributes were converted to categorical attributes, so there is no need for such an operation in this study.

### 3.2.1.3. C4.5 Decision Tree

C4.5 is a classification type decision tree algorithm that uses training set to build the model and make predictions on test instances. To construct C4.5 decision tree, information gain is used as the splitting criteria of the selected attributes. The tree consists of nodes which are the selected attributes and of edges for splitting the values of chosen attributes. Attributes with the highest information gain are chosen as nodes (splitting attributes). This process continues in a recursive manner until there is no improvement of information gain or the leaf nodes contain the instances having the same class value. The hypotheses space for this learning algorithm consists of candidate decision trees constructed from the same training set. This learning algorithm prefers short trees over longer and complex trees. (Quinlan J. , 1993) C4.5 is the extension of an earlier version of the decision tree, ID3 (Quinlan J. , 1986), developed to improve the ID3 tree algorithm. (Witten, Frank, & Hall, 2011) (Mitchell T. M., 1997) In C4.5 issues like missing values, numerical attributes, computational complexity and costs, and overfitting have been handled. (Mitchell T. M., 1997) (Witten, Frank, & Hall, 2011) Pruning is the method of cutting the branches of the tree to avoid overfitting (Shmueli, Patel, & Bruce, 2010) which is caused by the noise in the data or the inadequacy of the training examples (Mitchell T. M., 1997), by regarding the classification error, so in this study pruning method is used for decision tree algorithm.

In the article written by (Quinlan J. , 1986), a small training set about the decision of playing tennis on Saturday mornings was used to explain ID3 decision tree algorithm. This dataset includes 14 instances and 4 attributes named as outlook, temperature, humidity, and windy. The set of possible values for these attributes are as follows:

- Outlook: {sunny, overcast, rain}
- Temperature:{cool, mild, hot}
- Humidity: {high, normal}
- Windy: {true, false}

The class attribute for this dataset is binary and has the values as “P” for the Saturday morning being suitable for playing tennis and “N” for the decision of not to play tennis. The following table presents this sample data used to construct the simple tree given in (Quinlan J. , 1986).

Table 1- The small sample dataset taken from the original work of (Quinlan J. , 1986)

| No. | Attributes | | | | Class |
|---|---|---|---|---|---|
| | Outlook | Temperature | Humidity | Windy | |
| 1 | sunny | hot | high | false | N |
| 2 | sunny | hot | high | true | N |
| 3 | overcast | hot | high | false | P |
| 4 | rain | mild | high | false | P |
| 5 | rain | cool | normal | false | P |
| 6 | rain | cool | normal | true | N |
| 7 | overcast | cool | normal | true | P |
| 8 | sunny | mild | high | false | N |
| 9 | sunny | cool | normal | false | P |
| 10 | rain | mild | normal | false | P |
| 11 | sunny | mild | normal | true | P |
| 12 | overcast | mild | high | true | P |
| 13 | overcast | hot | normal | false | P |
| 14 | rain | mild | high | true | N |

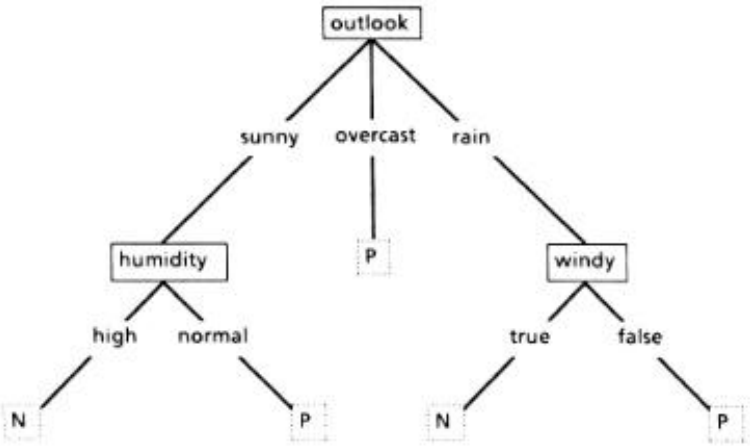



Figure 1- A sample decision tree taken from the original work of (Quinlan J. , 1986) and constructed from the Saturday mornings dataset

The sample decision tree given at the figure above is one of the candidate hypotheses to be searched over and is constructed by using training data using the information gain calculations. The calculation of the information gain is done based on probabilities of class values proportional to whole instances belonging a specific attribute value (a branch). Once the tree is constructed, it can be utilized to classify test samples. Testing starts from the root node placed at the top of tree and goes through the sub-tree until reaching a leaf node. To clarify the process, a test sample could be thought having the following values in its feature set:

Outlook, Temperature, Humidity, Windy = {rain, mild, normal, false}

Test starts with the root node of the learned tree above. The test instance has the value of "rain" for the root node, outlook attribute, so for the next step the direction will be the most right branch towards the child node, windy. This node checks the test sample's value for windy, and because it has value of "false", selected branch of this node brings the instance to leaf, class value, labeled as "P". As a result, for such a Saturday morning having temperature as mild, with normal humidity, rain and no wind, playing tennis would be preferred.

C4.5 algorithm can be applied on both numerical and categorical variables and handle missing values. (Quinlan J. , 1993) Hence it does not require the preprocessing of data points. Despite requiring many data points to be constructed, C4.5 is computationally efficient and can be applied for large datasets after construction. (Koshal & Bag, 2012) (Shmueli, Patel, & Bruce, 2010) In addition, it provides a schematic representation which can easily be interpreted by users. (Shmueli, Patel, & Bruce, 2010) Also, the selected decision trees can be converted to a set of rules to make them more readable. (Witten, Frank, & Hall, 2011) In this study, J48, the implementation of C4.5 in Weka is used to construct a classification tree and then prune it.

#### 3.2.1.4. Random Forest

Random forest is an ensemble method which is constructed by several decision trees to vote on the classification (or regression) task and developed by (Breiman, 2001). The trees constructing the forest have impact on the response. (Horning, 2010) The main idea of the random forest, like other ensemble methods, is that weak learners come together and by joining their power, a stronger model giving better results is formed. (Horning, 2010) In random forests, splitting attributes are chosen randomly, so the correlation between trees is decreased resulting in improved prediction accuracy. In random forest algorithm, the number of trees and the features to construct trees are chosen by the user. However while training random forest in Weka, there is no necessity to choose the number of features, Weka fulfills this job by considering some function background. Hence, the default settings for random forest is used which also defines the number of trees as 10. Random forests improve the results of decision trees owing to their voting among the trees, random feature selection, and bagging technique used to construct training samples for each tree. Bagging technique means to construct new training sets from the original training set with replacement sampling, i.e. without removing the selected samples from the original set. To measure the generalization error, the internal error estimate of the algorithm, out of bag error is used. The idea underlying out of bag error is to use the bagged training sets (the data which are not used to construct the trees) to make out of bag estimations and then calculate their prediction errors. (Breiman, 2001). They handle the overfitting problem, the sensitivity to outliers of decision trees, and so eliminate the need for pruning. ( Ali, Khan, Ahmad, & Maqsood, 2012) Overfitting is avoided by using the Strong Law of Large Numbers, because random forests always converge. (Breiman, 2001). Additively, they are fast and can handle missing values like decision trees and asymmetrical data in terms of class values. (Benyamin, 2012) Below a representative figure for random forests can be seen:

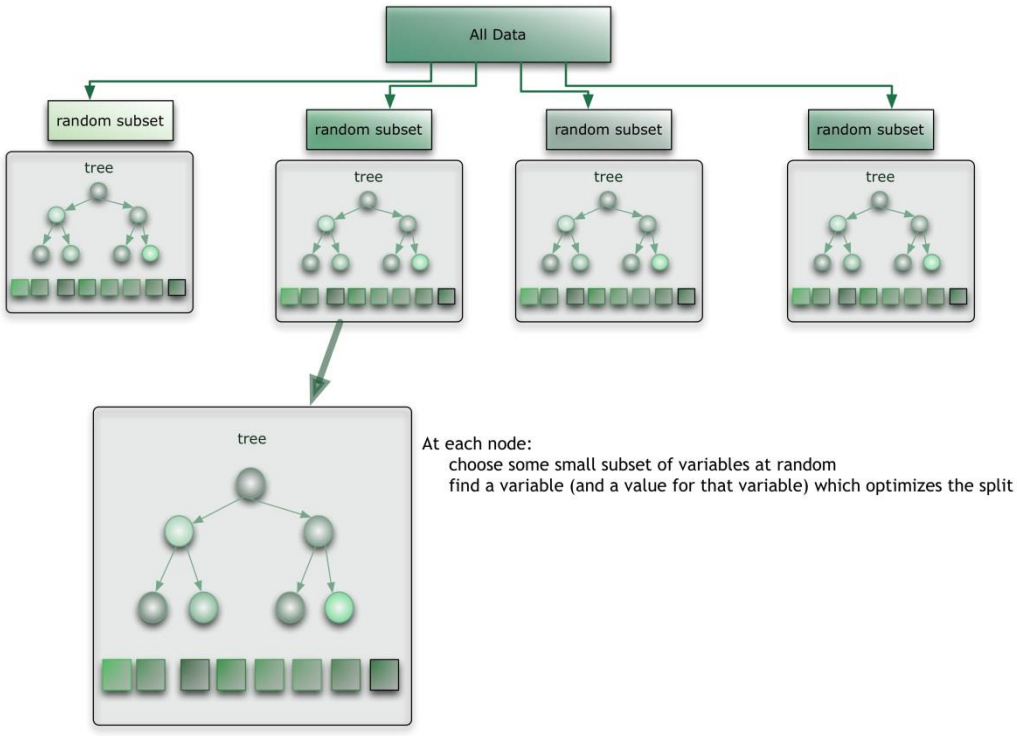



Figure 2- Random Forest representation (Benyamin, 2012)

In the above figure, the mechanism of random forests is demonstrated. The whole training set is split into several sub training set in a random way with replacement sampling as explained above. Then, decision trees are constructed for each training sets by using the random subset of features. In each tree, the optimum split is found by using those random feature subsets as search space and information gain measure. The above figure explains the construction of trees in a forest, but the process is not complete. After constructing trees, test samples are classified by each tree according to a decision tree classification procedure. Then, the final decision on the class of a test sample is given by voting among trees in the forest.

### 3.2.2. Feature Selection Algorithms

When working with a dataset with high dimensionality, as in this study, some of the attributes which are not relevant and necessary to build the model can cause overfitting and so reduces the generalizability of the algorithm. Also called as "curse of dimensionality", this problem may mislead the algorithm, increase the computational complexity and time to complete running the algorithm. ( Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012) Hence the feature selection algorithms have three goals as to enhance the performance of the learning algorithm, to obtain a faster model by reducing the computational cost, and to achieve obtaining a deeper comprehension about the underlying processes. (Yu & Liu, 2003) (Bai, Wang, & Zou, 2014) To overcome these problems in this study, feature selection methods are applied on the dataset by reducing the dimensionality.

Feature selections algorithms are categorized into two main types as *filter* and *wrapper* approach according to the implementation way and dependence on class values. In the wrapper

approach, to select relevant features, the target learning algorithm is used. In this approach, the searching algorithm solves optimization problem on the predictive accuracy of the chosen learning algorithm by searching through the space of feature subsets. (Cunningham & Delany, 2007) However in filter approach, attributes are selected before applying a learning algorithm independently by evaluating a predefined criterion like t-test, $\chi^2$-test and information gain. (Bai, Wang, & Zou, 2014) Filter approach has advantages of being fast, generalizable and unbiased from any classifier. ( Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012) Additionally, a former study comparing several feature selection methods of filter type is mentioned by ( Geng, Liu, Qin, & Li, 2007) to find Chi-Square and Information Gain as the most effective methods. ( Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012) also uses Information Gain and Chi-Square methods to select most relevant features for conducting an anomaly detection study on mobile devices with Android OS. Hence, in this study *Chi-Square* (Imam, Michalski, & Kerschberg, 1993) , *Information Gain* (Yang & Pedersen, 1997) and *ReliefF* (Kononenko, 1994) algorithms are selected as filter type feature selection algorithms. Since in the filter approach the number of features to be selected is required from the user, 10, 30 and 50 features are decided to be selected to run feature selection algorithms.

#### 3.2.2.1. ReliefF Feature Selection Method

ReliefF algorithm (Kononenko, 1994) is the extension of Relief algorithm (Kira & Rendell, 1992) which is limited to the classification problems with two class values. Unlike Relief, ReliefF handles multi valued class attributes and noisy data. ( Sikonja & Kononenko, 2003) The logic of the ReliefF is very similar to kNN algorithm. The closest data points around of an instance are expected to have similar attribute values with the instance questioned if they are the members of same class. Those attributes are the relevant ones that the algorithm tries to find out. (Yang & Li, 2006) The algorithm works as follows: A random instance is selected, and then nearest-hit (the nearest points having the same class value with the random point) and nearest-miss (the nearest points having the different class value with the random point) points are found. To find the relevant features, a parameter is calculated for each feature by finding the difference of the regarded attribute value between the chosen instance and the closest data points. If the feature under investigation has a high relevance, it means that it separates different classes successfully, then the difference parameter should be high for this relevant feature. (Kira & Rendell, 1992) ( Sikonja & Kononenko, 2003) (Yang & Li, 2006)

#### 3.2.2.2. Chi-Square Feature Selection Method

The statistical $\chi^2$ test is used to test the null hypothesis whether or not two variables are independent by evaluating the correlation between them. (Thabtah, Eljinini, Zamzeer, & Hadi, 2009) ( Uysal & Gunal, 2012) (Novakovic, Strbac, & Bulatovic, 2011) Hence, in terms of feature selection task, Chi-square test means testing the independency of an attribute to the class values. The $\chi^2$ formula below is adopted to feature selection problem calculating the $\chi^2$ score for each feature with respect to class value. It uses the observed ($N_{t,C}$) and expected ($E_{t,C}$) occurrences of specific feature values ($t$) for the instances regarding the class value ($C$). t's and C's having two values as 0 and 1 means that they represent the instances having/not having that specific attribute value and class value. ( Uysal & Gunal, 2012) (Thabtah, Eljinini, Zamzeer, & Hadi, 2009) Higher values of $\chi^2$ requires the rejection of null hypothesis, (Novakovic, Strbac, & Bulatovic, 2011) then it means that the feature being evaluated is relevant to the class values. ( Vryniotis, 2014)

$$\chi^2(t,C) = \sum_{t\in\{0,1\}} \sum_{C\in\{0,1\}} \frac{(N_{t,C} - E_{t,C})^2}{E_{t,C}}$$

The $\chi^2$ values for each feature are calculated in this manner and the features with high $\chi^2$ values rejecting the null hypothesis are chosen as candidate features. Among candidate features, ones with the highest score are chosen according to predefined number of features by ranking. ( Vryniotis, 2014)

### 3.2.2.3. Information Gain Feature Selection Method

Information Gain measure as a feature selection method is used to comprehend the contribution of a feature to classification task by calculating the entropy gain when that feature used as predictor. (Jamali, Bazmara, & Jafa, 2012) ( Uysal & Gunal, 2012) Below is the formula used to calculate information gain:

$$IG(t) = -\sum_{i=1}^{M} P(C_i)logP(C_i) + P(t)\sum_{i=1}^{M} P(C_i|t)logP(C_i|t) + P(\bar{t})\sum_{i=1}^{M} P(C_i|\bar{t})logP(C_i|\bar{t})$$

P($C_i$) is the probability of class values, P(t) and P($\bar{t}$) represents the presence or absence of a feature value, P($C_i|t$) and P($C_i|\bar{t}$) are the conditional probabilities of class values given the presence or absence of the regarded feature value. ( Uysal & Gunal, 2012)

### 3.2.3. Summary for Selected Algorithms

In this study, Naïve Bayes, k-nearest neighbor, J48 (java implementation of C4.5 decision tree), and random forest classification algorithms were chosen for Android malware detection by using official market metadata as predictor. In addition, as it is explained in "Data Preprocess and Features for Machine Learning Algorithms" part, the dataset used includes 861 features which may include irrelevant and unnecessary attributes for classification task. This may cause overfitting, mislead the learning algorithms, and increase the complexity and time needed to complete the algorithms. In order to mitigate this, feature selection methods were applied and the most relevant features selected by these methods were given as input to classification algorithms.

The reason for selecting Naïve Bayes as a supervised learning algorithm in this study is mainly its being a simple approach and a fast algorithm due to low computational complexity. (Witten, Frank, & Hall, 2011) (Shmueli, Patel, & Bruce, 2010) Despite its simplicity, it works well in practice and produces better or at least the similar results with more sophisticated algorithms. (Witten, Frank, & Hall, 2011) Similarly, as another simple and effective method, kNN was chosen. (Witten, Frank, & Hall, 2011) (Bhatia & Vandana, 2010) kNN owes its simplicity to its training phase which only stores training examples and does nothing more in terms of learning. Because of this advantage, new instances can be added to training set anytime (Witten, Frank, & Hall, 2011) as another pro. The estimation of target function is made for each new test sample locally and not for the whole dataset once. (Mitchell T. M., 1997) C4.5 was preferred because it is computationally efficient, (Koshal & Bag, 2012) (Shmueli, Patel, & Bruce, 2010) easy to

interpret with the help of its schematic representation and rules presented (Witten, Frank, & Hall, 2011) (Shmueli, Patel, & Bruce, 2010). Also because of handling missing values, both numerical and categorical data (Quinlan J. , 1993), it does not require to preprocess the data. Random forest learner (Breiman, 2001) is formed by several decision trees and has voting mechanism among those trees. Hence, they are computationally efficient and handle missing values like decision trees. Additionally, random forest handles the overfitting problem a decision tree faces, and eliminates the need for pruning. (Breiman, 2001).

For feature selection algorithms, filter approach is adopted in this study because it is fast, generalizable and unbiased from a learning algorithm unlike wrapper approach. (Yu & Liu, 2003) ( Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012) The chosen Chi-Square, information gain and reliefF algorithms are the common feature selection algorithms under filter approach, hence they were preferred in this study. Also, the use of Chi-Square and information gain algorithms was motivated by the study of ( Shabtai, Kanonov, Elovici, Glezer, & Weiss, 2012) which implements a real-time dynamic Android malware detection by comparing the combination of several classification and feature selection algorithms, and the number of features selected.

Other classification and feature selection algorithms instead of the chosen algorithms in this study could have been preferred, but the selected algorithms were accepted as working assumption. Implementation of different classification and feature selection algorithms for the Android malware detection method proposed in this study is considered as the future work.

## 3.3. Data Collection

The aim of this study is to investigate whether or not Google Play, Google's and Android's official application market, metadata of Android applications contribute to explain the malicious behaviors when combined with the analysis of user permissions. In order to achieve this goal, metadata of applications on Google Play were needed to be collected. First, a kind of web automation and testing tool and a browser-based macro recorder, iMacros was used to collect Google metadata and class (or target) values of Android applications. (iMacros) The data collection processes were recorded and the macro codes of these processes produced by iMacros were embedded into Microsoft Visual Basic in order to perform repetitive data collection tasks. However, because of requiring much amount of time for retrieving tens of thousands of applications' data by clicking one by one, it turned into a burdensome task. Then a more practical solution to this problem, using a web crawler and querying data directly from the servers of Google Play was preferred. The Google Play Crawler was used for acquiring requested permissions by applications and for downloading them as in the format of apk file. (Demiröz, 2013) Permissions collected for the top free applications in each application category consist of 851 different permissions in total, some of which includes developer-defined permissions. This crawler does not give all of the information presented to a user when he visits the Android application's page on Google to download. So, a java application was implemented to collect the other metadata of applications apart from permissions. Applications were downloaded and their Google Play information was gathered on 17 June 2014. Approximate total time for data collection was 12 hours The dataset contains top free applications from each application category and totally 17244 applications on the date of collection. The mentioned Google metadata of applications throughout this study includes the following information:

- Application category: Indicates the category of an application as defined by official market. There are two main categories as application and game. Full list of categories can be found at Appendix A
- Developer name: The name of an application developer on the official market. Developer names can be personal names or company names, if the application is developed by a company, or can belong to an institution like metropolitan municipalities which presents applications for transportation services.
- Developer type: Represents whether an application is developed by a top developer on official market or not or it is a type of editors' choice application. Top developers are promoted by Google and this is utilized by developers for advertisement and financial gain, by users for finding more trustworthy applications. Editors' choice applications correspond to the applications as some of the best applications chosen by Google Team. (Google, 2014) This attribute has three values for the dataset used in this study: Top developer, Editors' Choice Top Developer, Editors' Choice.
- Star 5, Star 4, Star 3, Star 2, Star 1 rating counts: Display the count of stars for each star number (1, 2, 3, 4, 5) given by users who download an application. There is no lower limit to show star ratings for an application on its page. All of star ratings counts would be zero if any user have not rated an application yet and will be displayed as zero for all star ratings (1,2,3,4,5) on the application's page. In addition, ratings presented on an application's page on Google are valid for all of the versions of it, i.e. they are lifetime ratings for an application.
- Average rating of application: Derived from star rating counts of an application, it is counted as weighted average of star ratings. Average rating can take 100 as maximum value and 0 as minimum because an application may not be rated yet by any user. In fact, the dataset used in this study includes such applications which have 0 star ratings for all star levels and have 0 for average rating.
- Publish date of the current version of application: Developers can update their applications presented on the market and so the version number of them. Then they can upload those new versions of their applications on market. The label "Updated" at the bottom right area of the application's page displays the publish date of applications most recent version.
- Size of application: This is a numeric attribute and contains values starting from 11 to 994,304 bytes for the dataset used in this study. However, for some applications this area has "varies with device" value, hence discretization was applied as these applications would fall into one category.
- Minimum required Android OS version: When developing Android applications, developers are asked for the minimum required level of Android API. Android API is an integer number and is used to compare the compatibility of an Android application with Android platform on a mobile device. If a user's mobile device hosts a system with a lower API level than the API level stated as the minimum required for an application to run on a device, then the regarded application would not run on the user's device. (developer.android.com, 2014)
- Content rating of application: Developers are required to choose proper content rating of their applications while uploading them on the Google Play. Categories for content rating in the dataset used in this study are "everyone, low maturity, high maturity, medium maturity, and not rated" and Google has guideline for selecting them according to the level of harm (like for gambling, violence, hate, alcohol, tobacco and drug, sexual content) can be caused by content. (Google, 2014)

- Download number of application in form of range: It shows the total install number of an application by users as an interval (e.g. 10,000-50,000) Full list of download ranges can be found at Appendix A.

A sample interface for an application on Google Play can be seen at the following figures:

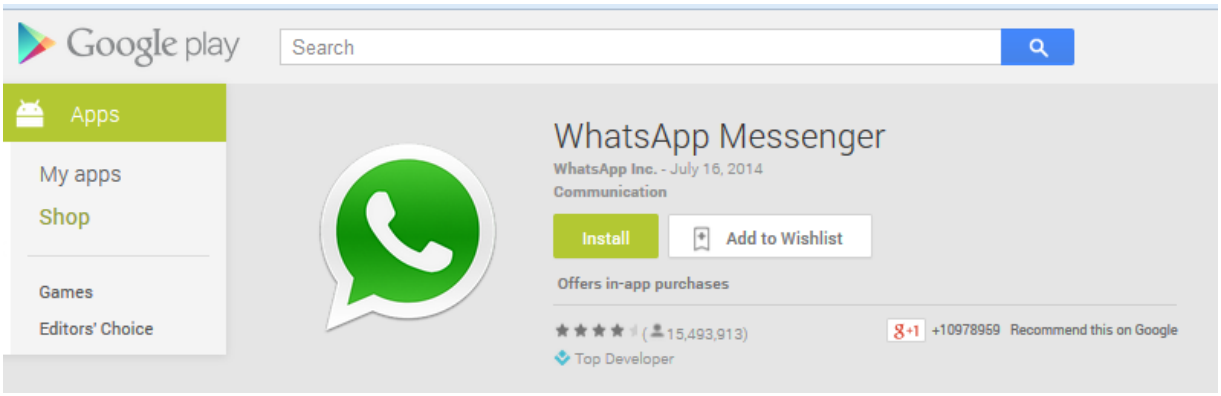



Figure 3- Developer name, application type and developer type information of the sample application

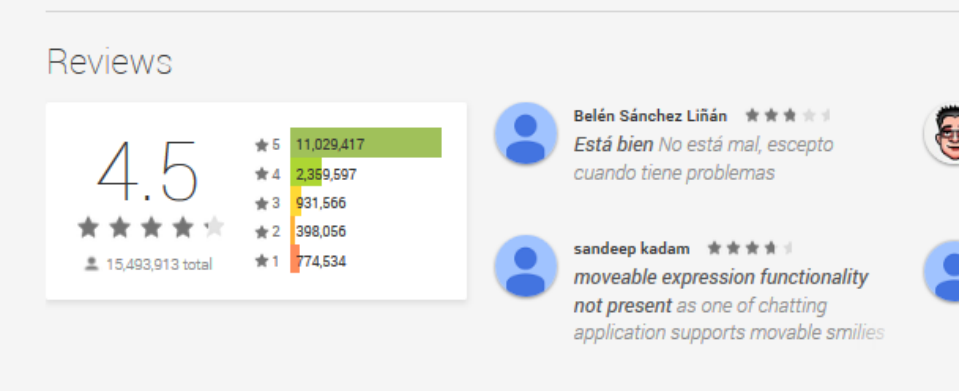



Figure 4- Average rating and star ratings of the sample application

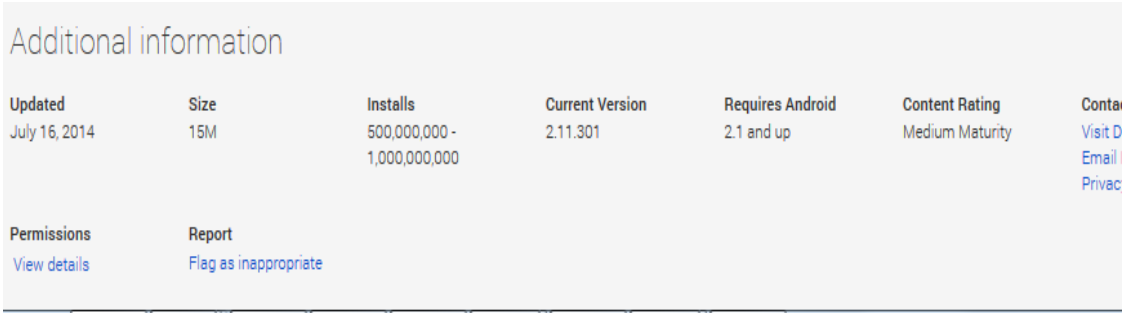



Figure 5- update date, size download range, minimum required Android OS, content rating of the sample application

After collecting market-related information of applications (including permissions), it was the second step to acquire the target values of applications, as malicious or benign, to be able to train a supervised classification algorithm. For this purpose, Virus Total, a free online service identifying malicious content with the contribution of several antivirus engines and web scanners was appealed. (VirusTotal) Virus Total enables searching different malicious content, like webpages by querying URLs, domains or IP addresses and files in many diverse format (including apk files) uploaded by the user to the VirusTotal or queried by using the hash values of files. To scan files by using hash values requires less time than uploading apk files to the web site one by one, so by using the "shortcut" (VirusTotal) provided by VirusTotal and sending http requests, the most recent reports of applications were obtained with hash values. According to the analysis results of antivirus engines, three Virus Total-related attributes were added into dataset:

- Analysis date of an apk file
- Number of AV engines which label the sample apk file as malicious
- Total number of AV engines which analyze the sample apk file

The first two data above was used to derive features for machine learning tasks, so the last one was disregarded.


Figure 6- Virus Total homepage

After this, a question arose how the hash values of the Android applications appearing in Google Play can be found. As the first option, "APK Downloader" website was utilized to query the md5 hash values of applications by using package names obtained in the first step. (APK Downloader) Using this website is an easy way to get the md5 values of Android application files and to download them; but the site imposes a quota by user and daily quota

which does not permits the generation of new apk files more than 1400. The author(s) of the website proposes the use of Chrome extension of APK Downloader; however it only downloads apk files, does not present md5 checksums. Hence, a second option to get the hash values of apk files had to be implemented: writing a Java application which uses the proper methods in Java by putting apk files into hash function and yielding fixed size hash values. Not only md5 checksums, but also sha256 checksums were calculated in order to validate the applications which analysis results obtained from Virus Total are the same with the ones downloaded from Google Play, because md5 checksum is questionable that some files may share the same md5. (Wikipedia, 2014) Sha 256 is stronger than md5because of having 256 bits length whereas md5 is 128 bits in length and is used by VirusTotal when showing analysis results. To sum up, md5 and sha256 hash values of each application were calculated, the detection results of those applications were queried by using md5 hashes and then the sha256 hashes given at the detection result pages of Virus Total and the calculated ones were compared to ensure similarity of apk files.

## 3.4. Data Preprocess and Features for Machine Learning Algorithms

In this study, to perform machine learning related tasks, Weka, open source data mining software in java, was used. ( Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) It allows preprocessing data, training and testing supervised/ unsupervised learning algorithms, visualizing data, conducting feature selection on attributes and experiments for comparison on the results of different algorithms, and obtaining learning curves etc. Raw data collected from market could not be used directly in Weka to apply the feature selection and classification algorithms selected. First of all, some instances of applications had negative numbers in malware detection count field (this feature corresponds to the number of AV engines which identified the application as malware) by fault in collection process; these 12 instances were excluded from dataset. Similarly, one instance had a date value in the size of application column, so this one was omitted too. Finally, a dataset consisting of 17,321 instances was obtained.

There were 2 attributes in the dataset holding instances with values in type of date, namely analysis date of the application which was obtained from Virus Total and the publish date of the application's current version which is presented on the Google Play web page. This raw information does not add meaningful information solely, so they were converted to days passed after detection and publish date and two continuous variables were obtained. Additionally, some of the developer names of applications on their Google Play webpage contain non-Latin characters, like Cyril or Japanese. The developer names were changed with the ones in the format of D1, D2, D3, and so on, up to 9321$^{st}$ developer because Weka cannot process this kind of variables. In addition, star rating count information of applications was omitted from the scope of the analyses because average rating of applications is calculated by the weighted average of these numbers. Another arrangement on the data was to convert size of all applications into bytes for the purpose of measuring them in the same unit. Any of the remaining variables which are application category, developer type, minimum required Android OS version, content rating, and the download range of the application was not subjected to modification process because they had the standard and finite number of categories. Lastly, the two Virus Total-related attributes, the number of AV engines which label the apk files as malicious and the total number of AV engines which analyze the apk files, were handled. The latter one was ruled out of the study because it does not contribute to the settings of the current study. The former one was used to label instances as "malware" or "benign applications". As it

will be discussed in the baseline datasets section in detail, different labeling methods were experimented by changing the number of detection times.

After first arrangements on dataset mentioned above, discretization was applied on continuous attributes (average rating of application, size of application, days passed after detection date and days passed after publish date) in order to be able to train Naïve Bayes classification algorithm. Discretization corresponds to lessen the number of values a continuous variable takes by dividing the variable into a number of intervals. (Joita, 2010) There are two main types of discretization algorithms as unsupervised which does not require class information to perform discretization task and supervised which takes into account the class labels. (Dash, Paramguru, & Dash, 2011) In this study, equal-frequency binning, one of the two classical unsupervised discretization methods (the other one is equal-width), (Al-Ibrahim, 2011) is chosen because "Unsupervised discretization algorithms are the simplest to use and implement. They only require the user to specify the number of intervals and/or how many data points should be included in any given interval." (Cios, Pedrycz, Swiniarski, & Kurgan, 2007) (p. 237) Moreover, equal-width discretization method may cause the loss of information after discretization if the variables to be discretized are not evenly distributed. (Kotsiantis & Kanellopoulos, 2006) The handled continuous attributes in this study are skewed and not evenly distributed, so the equal frequency method is chosen. For seeing the distribution of those attributes, histograms were drawn and can be found at Appendix B.

Equal frequency binning method divides the attributes into intervals after sorting them in a way that each interval contains approximately the same number of data points. The number of intervals are defined by user and there is no a proven optimum way of doing this job. (Mitov, Ivanova, Markov, Velychko, Stanchev, & Vanhoof, 2008) However, three different approach were considered while defining number of intervals; Juran's rules for number of bins, square root rule and Freedman–Diaconis' choice. According to Juran, if the number of data points is more than 1000, then it may be suitable to select the number of bins between 11 and 20. (QIMacros) Square root rule simply takes the square root of number of data points and round this number up to find the number of intervals. Lastly, Freedman-Diaconis' choice applies the following formula to calculate the bin width:

$$h = 2\frac{IQR(x)}{n^{1/3}}$$

where the IQR(x) represents the inter quartile range of related variable and n represents the number of data points. Then by subtracting the minimum value of the regarded variable from the maximum value and dividing this range with the bin width, the number of intervals is calculated. (Wikipedia, 2014)

Using Freedman-Diaconis' rule produced very close results to the square root rule in terms of the number of bins for days passed after detection date, days passed after publish date of an applications' current version and average rate of application. For size of application in bytes attribute, Freedman-Diaconis' rule produced approximately 1350 bins, which can be considered as too many and the original logic of discretization disappears. As a result, giving the approximate results with Freedman-Diaconis' rule, square root rule was chosen to define the number of intervals in order to use in equal frequency discretization method.

Finally, a dataset was obtained consisting attributes which are all nominal attributes. This final dataset includes the following features:

- Application category (nominal attribute, having 43 sub categories)
- Content rating (nominal attribute, having 5 sub categories)
- Android version (nominal attribute, having 22 sub categories)
- Download number of an application (ordinal attribute in format of intervals, having 19 sub categories)
- Developer name of an application (nominal attribute, having 9321values)
- Developer type of an application (nominal attribute, having 4 sub categories)
- Size of an application in bytes (ordinal attribute, having 130 sub categories)
- Current average rating of an application (ordinal attribute, having 132 sub categories)
- Days passed after publish date of last version (ordinal attribute, having 130 sub categories)
- Days passed after detection date (ordinal attribute, having 130 sub categories)
- 851 Android permissions (binary attributes, having "yes" for the owned permissions by applications, "no" for the not included ones)
- Target attribute (class variable in binary format, having "benign" value as negative, and "malicious" value as positive samples)

## 3.5. Parameters Used to Evaluate Classification and Feature Selection Algorithms

In machine learning studies, the performance of proposed models is evaluated by utilizing some well-known metrics. After running classification algorithms, a matrix named as confusion matrix is obtained including the actual and predicted number of instances with respect to class values. The representation of a sample confusion matrix is provided below: ( Mitchell M. , 2010)

Table 2- Sample Confusion Matrix

| | | Predicted class | |
|---|---|---|---|
| | | Positive | Negative |
| Actual class | Positive | True Positive | False Negative |
| | Negative | False Positive | True Negative |

In this study, malicious applications are accepted as positive instances and benign applications as negative ones. Hence, the cells of the confusion matrix for this study can be interpreted as follow:

- True Positive (TP): Number of malicious applications which correctly classified, a.k.a. hit
- False Positive (FP): Number of benign applications which are incorrectly classified as malicious, a.k.a. false alarm
- True Negative (TN): Number of benign applications which correctly classified, a.k.a. correct rejection, a.k.a. miss

- False Negative (FN): Number of malicious applications which are incorrectly classified as benign

These values provided by the confusion matrix are used to calculate performance measures for the assessment of classification algorithms. The following measures derived from confusion matrix (except kappa statistic) are adopted in this study: (Wikipedia, 2014)

- Accuracy (ACC) : Rate of correctly predicted applications to the total number of applications

  ACC= TP+ TN / (TP+ TN+ FP+ FN)

- True Positive Rate (TPR) : Rate of correctly predicted malicious applications to the total number of malicious applications , a.k.a. recall rate

  TPR= TP/P = TP / (TP+FN)

- False Positive Rate (FPR) : Rate of incorrectly predicted benign applications as malicious, to the total number of benign applications

  FPR= FP/N = FP / (FP+TN)

- True Negative Rate (TNR) : Rate of correctly predicted benign applications to the total number of benign applications

  TNR= TN/N = TN / (TN+FP)

- False Negative Rate (FNR) : Rate of incorrectly predicted malicious applications as benign, to the total number of malicious applications

  FNR= FN/P = FN / (TP+FN)

- Precision : Rate of correctly predicted malicious applications to the total number of predictions as malicious, a.k.a. positive predictive value

  Precision= TP/ (TP+ FP)

- Kappa statistic: This statistic is used to examine the compliance of predictions according to class values. If the class values (benign, malicious) are in complete agreement then the kappa statistic equals to 1 and the worst case is denoted by 0, so the higher values of kappa statistic are preferable. (Abela, Angeles, Delas Alas, Tolentino, & Gomez, 2013)

## 3.6. Pilot Study and Baseline Datasets

As the first step of the analyses, a prior study was carried out on the dataset including the whole applications collected from the market (17,231 preprocessed applications). Since the dataset includes too many observations, in order to obtain the results in a fast manner for a pilot study, a classification algorithm with low computational complexity was needed and as mentioned in the classification algorithms part, Naïve Bayes meets this requirement. If acceptable prediction

results had been obtained from Naïve Bayes classifier, then the conclusion would be drawn that other classifiers would probably give similar results in terms of evaluation parameters.

There were 4,512 malicious and 12,719 benign applications in this initial dataset. The target values were defined according to the detection count data collected from Virus Total. For this initial set, applications having detection count greater than or equal to 2, i.e. which have been identified as a malware by at least 2 AV engine were labeled as malicious applications. The remaining applications were labeled as benign applications. In order to analyze the contribution of other Google market metadata to the permissions of Android applications, two datasets were constructed:

- One including only permission information
- The other including permissions+ other Google market metadata

Using 10-fold cross validation, Naïve Bayes classification algorithm for these two dataset was run 10 times by changing the random number seed for every run used to split dataset. The average results according to runs are shown below:

Table 3-Results of Pilot Study

| Classifier | Feature Set | Accuracy | TPR | FPR | Precision | Kappa Statistic | Class |
|---|---|---|---|---|---|---|---|
| Naive Bayes | Permissions | 72.35% | 0.85 | 0.63 | 0.79 | 0.24 | B |
| | | 72.35% | 0.37 | 0.15 | 0.47 | 0.24 | M |
| | Permissions + other Google Play metadata | 76.11% | 0.84 | 0.46 | 0.84 | 0.38 | B |
| | | 76.11% | 0.54 | 0.16 | 0.55 | 0.38 | M |

B denotes benign class value, M denotes malicious class value

This display format of parameters is the same Weka presents. For the ease of interpretation, one should take into account that in this study, the malicious samples are regarded as positive, benign ones are regarded as negative class values.

According to the results of initial analysis, when only permissions are used as predictors in Naïve Bayes classification algorithm, 72.35% accuracy is obtained. If the other metadata of applications collected from Google Play is added to the model, then the accuracy slightly increases to 76.11%. However, the addition of Google Play metadata seriously increases the TP rate (the percent of correctly detected malware samples) while decreasing FP rate (the percent of benign samples incorrectly identified as malware). The contribution of the added attributes can be realized clearly by comparing the kappa statistics. Despite the difference of accuracy values give rough idea about the improvement of classification algorithm, kappa statistic strengthens inferences. In other words, accuracy may increase but this improvement may be in favor of only one class, but kappa statistic takes into account of the agreement between different classes.

Another important inference from the results is a general problem about accuracy levels. These accuracy levels can be taught as acceptable, but not satisfactory. As it is stated in feature selection part, excessive number of features may mislead the learning algorithm and cause to decrease the accuracy. Hence, to see the impact of reducing the number of features, three chosen algorithms (chi-square, information gain and reliefF) were applied on the dataset including all the applications collected and the Google Play market metadata together with permissions. The number of features which would be selected by feature selection algorithms was identified as 50 for initial study, because the aim was not to tune the parameters for feature selection algorithms at this step. Again algorithms (Naïve Bayes with and without feature selection algorithms) were run for 10 times, but without cross-validation due to the fact that big datasets require high computation time for implementing cross validation. Instead 80% random split of the data was used for training set and the remaining 20% for testing set. The results can be seen at the table below:

Table 4- Results of feature selection algorithms on initial dataset

| Classifier | FS Algorithm | Feature Set | Accuracy | TPR | FPR | Precision | Kappa Statistic | Class |
|---|---|---|---|---|---|---|---|---|
| Naive Bayes | - | Permissions + other Google metadata | 75.97% | 0.84 | 0.47 | 0.84 | 0.38 | B |
| | | | 75.97% | 0.53 | 0.16 | 0.54 | 0.38 | M |
| Naive Bayes | IG | Permissions + other Google metadata | 75.68% | 0.83 | 0.46 | 0.84 | 0.37 | B |
| | | | 75.68% | 0.54 | 0.17 | 0.54 | 0.37 | M |
| Naive Bayes | CS | Permissions + other Google metadata | 75.65% | 0.83 | 0.46 | 0.84 | 0.37 | B |
| | | | 75.65% | 0.54 | 0.17 | 0.54 | 0.37 | M |
| Naive Bayes | RfF | Permissions + other Google metadata | 76.21% | 0.84 | 0.46 | 0.84 | 0.38 | B |
| | | | 76.21% | 0.54 | 0.16 | 0.55 | 0.38 | M |

B denotes benign class value, M denotes malicious class value
FS: Feature Selection, IG: Information Gain, CS: Chi-Square, RfF: ReliefF

When the results are examined, it is seen that the feature selection algorithms did not increase the accuracy of the model except ReliefF, but with a slight increase. In addition, FP rates and kappa statistics did not differ notably. To sum up, applying feature selection methods on this dataset did not improve the results as wished, so another different method was needed for increasing accuracy while decreasing or at least not increasing the false positive rate at the same time. The dataset was imbalanced in terms of class attribute, consisting of 74% benign and 26% malicious applications according to the defined detection number (by accepting the applications which are identified as malware by more than or equal to 2 AV firms). As a first choice, balancing the dataset was thought and the following 2 datasets were constructed as balanced ones:

- A dataset including the same number of (4,512) malicious and benign applications: Malicious applications were kept, only 4,512 benign applications were chosen randomly among 12,719 benign applications.
- A dataset including the same number of malicious and benign applications under each application category: For an application category, if there were more benign applications than malicious ones, benign applications were chosen randomly as their number became equal to the malicious ones. If the number of benign applications was smaller than or equal to the number of malicious applications in an application category, nothing was changed.

The table below shows the comparative results of these 2 datasets with the original dataset:

Table 5- Comparison of detection results for balanced and imbalanced datasets

| Classifier | Dataset | Feature Set | Accuracy | TPR | FPR | Precision | Kappa Statistic | Class |
|---|---|---|---|---|---|---|---|---|
| Naive Bayes | İmbalanced (original) | Permissions + other Google metadata | 75.97% | 0.84 | 0.47 | 0.84 | 0.38 | B |
| | | | 75.97% | 0.53 | 0.16 | 0.54 | 0.38 | M |
| Naive Bayes | # Malware and Benign equal | Permissions + other Google metadata | 68.36% | 0.71 | 0.34 | 0.67 | 0.37 | B |
| | | | 68.36% | 0.66 | 0.29 | 0.69 | 0.37 | M |
| Naive Bayes | # Malware and Benign equal for each app category | Permissions + other Google metadata | 73.67% | 0.83 | 0.39 | 0.75 | 0.45 | B |
| | | | 73.67% | 0.61 | 0.17 | 0.71 | 0.45 | M |

B denotes benign class value, M denotes malicious class value

Both of the datasets has not shown an improvement in terms of prediction accuracy, but balancing the number of malicious and benign applications under each application category improves the false positive rate while not decreasing the accuracy of the model seriously. In order to understand whether the contribution of false positive rate's decrease to the class agreement is statistically meaningful, a paired t- test on the kappa statistic of two datasets (the original one and the one having the same number of malicious and benign applications under each application category) was applied. According to the 5% confidence level, the balanced dataset has a higher kappa statistic than the unbalanced dataset which is statistically significant. As a result, this balancing option did not improve the accuracy but decreased the false positive rate significantly; hence it should be kept aside for further uses.

As the second option for getting more accurate results together with lower false positive rates, changing the labeling method of applications as malware or benign could be tried. In the first original dataset, applications were labeled as malicious or benign if their detection count is greater than or equal to 2. However, one cannot be so sure that the applications which are claimed to be malicious by 2 AV engines are malware actually. Similarly, the applications which are identified as malware by 1 AV engine may not be a benign application, but they were labeled as benign in the former method. To sum up, the detection count criteria for labeling applications as malware should be increased, and by only accepting the applications with zero

detection count as real benign ones, the remaining ambiguous applications should be eliminated from the dataset. In order to decide this detection count level, experiments were conducted by constructing datasets according to the principles mentioned above and applying Naïve Bayes classifier on those datasets. This time again the classification algorithm was run 10 times for each dataset by applying cross-validation because datasets got smaller and more suitable for cross-validation. The results of experiments are given below:

Table 6- Comparison of detection results for datasets with different detection counts of malware applications

| Classifier | Feature Set | Detection Count | Accuracy | TPR | FPR | Precision | Kappa Statistic | Class |
|---|---|---|---|---|---|---|---|---|
| Naive Bayes | Permissions + other Google metadata | 2 | 76.11% | 0.84 | 0.46 | 0.84 | 0.379 | B |
| | | | 76.11% | 0.54 | 0.16 | 0.55 | 0.379 | M |
| | Permissions + other Google metadata | 3 | 79.53% | 0.86 | 0.37 | 0.86 | 0.481 | B |
| | | | 79.53% | 0.63 | 0.14 | 0.61 | 0.481 | M |
| | Permissions + other Google metadata | 4 | 82.62% | 0.88 | 0.36 | 0.90 | 0.508 | B |
| | | | 82.62% | 0.64 | 0.12 | 0.62 | 0.508 | M |
| | Permissions + other Google metadata | 5 | 84.87% | 0.90 | 0.35 | 0.92 | 0.527 | B |
| | | | 84.87% | 0.65 | 0.10 | 0.54 | 0.527 | M |
| | Permissions + other Google metadata | 6 | 86.65% | 0.91 | 0.36 | 0.93 | 0.526 | B |
| | | | 86.65% | 0.64 | 0.09 | 0.58 | 0.526 | M |
| | Permissions + other Google metadata | 7 | 88.84% | 0.93 | 0.38 | 0.94 | 0.525 | B |
| | | | 88.84% | 0.62 | 0.07 | 0.57 | 0.525 | M |
| | Permissions + other Google metadata | 8 | 90.73% | 0.943 | 0.403 | 0.95 | 0.516 | B |
| | | | 90.73% | 0.597 | 0.057 | 0.54 | 0.516 | M |
| | Permissions + other Google metadata | 9 | 91.79% | 0.949 | 0.426 | 0.96 | 0.493 | B |
| | | | 91.79% | 0.574 | 0.051 | 0.51 | 0.493 | M |
| | Permissions + other Google metadata | 10 | 93.45% | 0.961 | 0.435 | 0.97 | 0.501 | B |
| | | | 93.45% | 0.565 | 0.039 | 0.51 | 0.501 | M |

B denotes benign class value, M denotes malicious class value

It is seen clearly that when the number of detection is increased in order to label an application as malware and the more ambiguous applications are removed, the prediction accuracy of the model increases constantly. However, as the detection count continues to increase, false alarms start to increase after some point because dataset becomes more imbalanced due to increasing proportion of benign applications. This can also be seen from the kappa statistics that the agreement between two classes deteriorates after detection count equals to 5. In order to choose the final baseline datasets, paired t-test was applied on prediction accuracies and kappa statistics of these datasets and can be found at Appendix C. All the increasing prediction accuracy seem to have statistically significant difference from the former ones, this t-test does not enlighten so much about the choice. Hence, when the t-test for kappa statistics is examined, the datasets which have detection counts 5, 6, 7 and 8 are found to have statistically significant difference. Among them, the dataset having the detection count as 8 is selected for further analysis because of two reasons. First, it has less data points from others and this helps to decrease the computation time. Second, the dataset with detection count equal to 8 is more accurate than the others (datasets with detection count 5, 6 and 7).

After the selection of the dataset with detection count=8, then the effect of first balancing method on the false positive rates and kappa statistic was questioned. The first method to balance the dataset was to choose the equal number of benign and malicious applications under each application category. Therefore, a new dataset had been constructed from the dataset which had detection count equals to 8 by leaving the same number of benign and malicious applications under each application category. Then Naïve Bayes classifier was run for 10 times with 80 percent random partition of datasets for training data and the left for testing data. Resulting prediction accuracies and kappa statistics were subjected to paired t-test to see whether the difference between these parameters for two datasets is statistically significant.

Table 7- Accuracy Comparison for balanced and imbalanced datasets with detection count=8

| Datasets / Classifier | Detection Count=8, balanced | Detection Count=8, imbalanced |
|---|---|---|
| Naïve Bayes | 80.65 | 90.74 v |

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

Table 8-Kappa statistic comparison for balanced and imbalanced datasets with detection count=8

| Datasets / Classifier | Detection Count=8, balanced | Detection Count=8, balanced |
|---|---|---|
| Naïve Bayes | 0.51 | 0.57 v |

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

According to results, when the dataset was balanced, the accuracy of the model decreased from 90.74 to 80.65 and the difference is statistically significant at 0.05 confidence value. However, the kappa statistic of the dataset increased from 0.51 to 0.57 and this difference is significant too. This means that the balanced dataset is less accurate but produces this accuracy value with higher agreement between benign and malicious class values and with less false positive rate. Consequently, the balanced dataset with detection count=8 was decided to continue with for further analysis and the final two baseline datasets were constructed:

- A dataset including only permissions as feature set, malicious applications which have been identified as malware by more than or equal to 8 AV engines and benign applications which have not been identified as malware by any of AV engines (and the number of benign and malicious applications under each application category was balanced as explained before in this section)
- A dataset including permissions together with other Google Play metadata as feature set, malicious applications which have been identified as malware by more than or equal to 8 AV engines and benign applications which have not been identified as malware by any of AV engines (and the number of benign and malicious applications under each application category was balanced as explained before in this section)

## 3.7. Summary of Pilot Study

Before applying all the classification and feature selection algorithms and their combinations with the number of selected features, a pilot study was carried out to see the approximate performance of the detection model proposed in this study. In order to obtain and compare results in a fast manner, Naïve Bayes classifier was preferred. As the first step, whole dataset of applications collected from the official market was used to apply Naïve Bayes learning algorithm. Applications' market related data was collected by writing a Java application for all application categories defined by Google Play and for top free applications under each of them (Those application categories can be found at Appendix A) . Permissions requested by applications at install time were collected by using a Google Play crawler (Demiröz, 2013) and applications were downloaded again utilizing this crawler. The reason for downloading applications is to calculate their hash values in order to query them via an online free AV engine (VirusTotal). After collecting applications, they were labeled according to the number AV engines detecting them as malicious. If an application had been identified as malicious by more than or equal to 2 AV engines, it was labeled as malicious, otherwise as benign, for the initial phase. Two different datasets were used to investigate the contribution of official market metadata; one of which only includes user permissions and the other includes user permissions together with market metadata. These two datasets included the same number of instances, consisting of 4512 malicious and 12719 benign applications. Naïve Bayes classifier was run 10 times for these two datasets with cross-validation. According to initial results, addition of market metadata increased the accuracy of model 3.76% which is a slight difference, but the amount of increase in TPR (0.165) and decrease in FPR (0.164) were acceptably high. This means that despite insignificant improve in prediction accuracy, addition of market metadata improved the agreement of class values (benign and malicious) on classification results.

However, the false positive rate was not acceptable enough yet and the prediction accuracy could be improved more. The dataset included 861 features, so as the effect of excessive number of features, classification algorithm could be misled. Therefore, 3 feature selection algorithms (chi-square, information gain and reliefF) are used to select the most important 50

features to give Naïve Bayes classification algorithm as input. This time, the dataset with proposed features was used by 80-20% split for training and testing sets. Results showed that applying feature selection algorithms had not improved the prediction accuracy, FP rates and kappa statistic notably. As another option to improve the results, balancing the dataset in terms of class values was applied. According to the classification results for the datasets obtained by balancing class values, balancing the number of malicious and benign applications under each application category does not improve the accuracy but improves the false positive rate.

As the last operation to improve the prediction accuracy, an experiment was conducted by changing the detection count used to label an application from 2 to 10. In this experiment, market metadata was used as proposed and the only applications which had not been identified as malicious by any AV engine were accepted as benign applications. The remaining ambiguous applications were excluded from the datasets. When the results are examined, it is seen that the dataset with detection count 8 best suits the needs with respect to prediction accuracy and kappa statistic. Then on this dataset, balancing operation was done for the number of malicious and benign applications under each application category. Despite obtaining a less accurate result with the balanced dataset, a higher kappa statistic was achieved meaning the higher agreement between class values and lower FPR. In addition, the resulting dataset is far smaller (approximately 25% of the former one) than the former one allowing classification and feature selection algorithms to require less time to be completed.

## 3.8. Schematic Representation of the Research Methodology

The figure below displays the processes of the research methodology applied in this study.

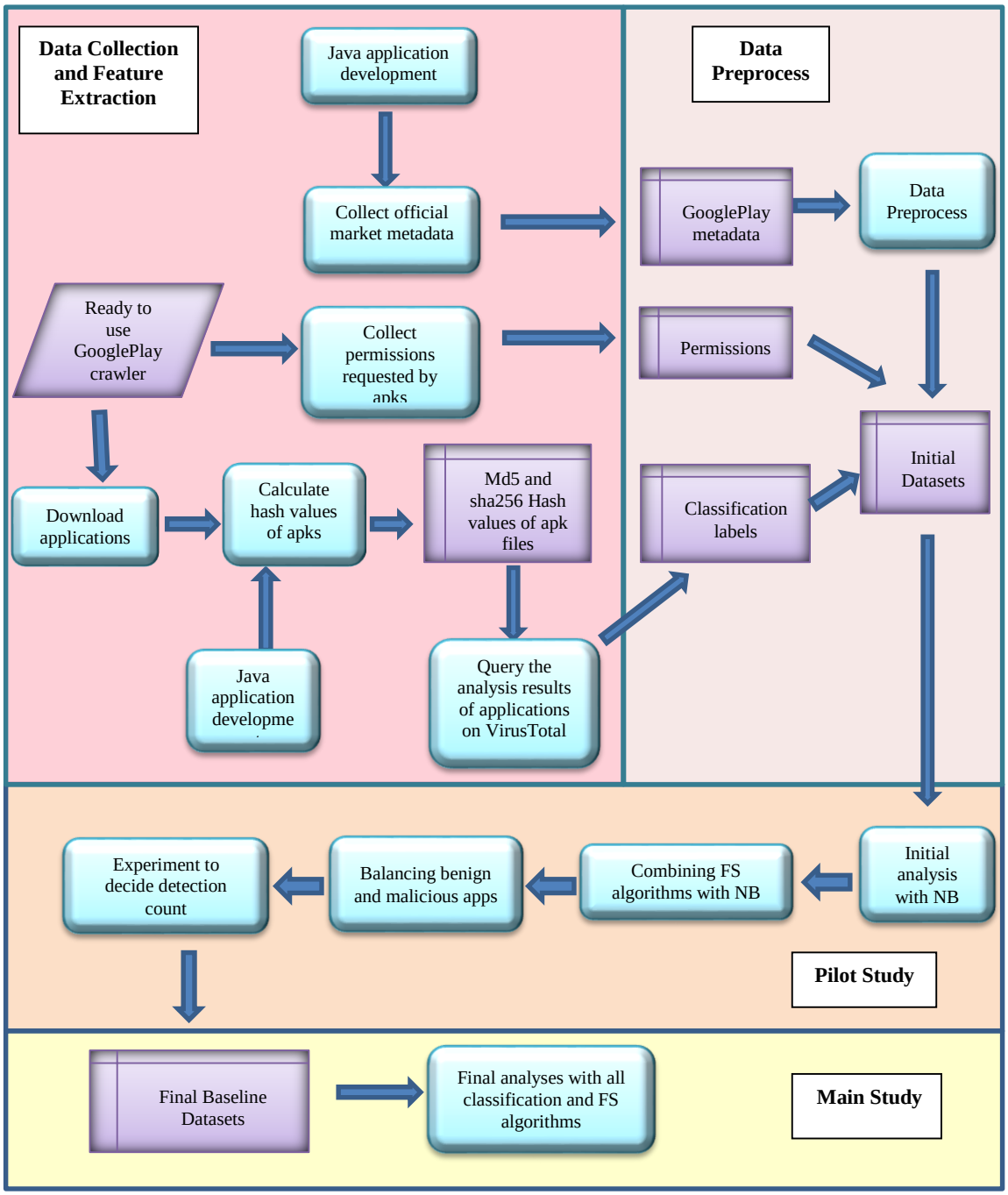



Figure 7- Steps for the research methodology of the proposed study

# CHAPTER 4

# RESULTS

In this chapter the evaluation of results for the classification and feature selection algorithms on the final baseline dataset is presented. First the contribution of official market on to the requested permissions from users is evaluated in terms of the prediction performance of machine learning algorithms. In addition, the selected classification algorithms are compared to find the most accurate algorithm. Lastly, the effectiveness of feature selection algorithms is evaluated and the best performing combination of classification and feature selection algorithms, and the number of selected features is attested.

## 4.1. Brief Information about Datasets and Configuration for Algorithms

In order to explore the contribution of other official market metadata of Android applications to the malware detection model which uses only requested permissions from users as feature set, 2 datasets were constructed in this study. The first dataset comprises only requested permissions and the second one has additional information about applications presented by Google Play Store. Those 9 additional features are listed in the previous chapter and detail information is given about them. Apart from those features, one more additional feature, the days passed after the detection date of a malicious application, was used as input for classification algorithms taken from VirusTotal website. These datasets include malicious applications which have been identified as malware by at least 8 or more AV engines on VirusTotal. Benign applications are the ones which have not been identified as malicious by any of the AV engines on the date of data collection. The remaining ambiguous applications were omitted from the dataset. As the last arrangement on the data collected from application market, the number of malicious and benign applications was balanced since it has been shown before in this study that balancing the dataset causes the decrease of false alarms. These two datasets includes 2528 instances 907 of which are malicious and the remaining 1621 ones are benign applications. Appendix D illustrates the histograms of applications produced by Weka for attributes related to Google market metadata.

Using two baseline datasets, 4 different classification algorithms and 3 different feature selection algorithms were run 10 times for each by randomly partitioning datasets. Cross-validation was not preferred because totally 80 algorithms were run together with combinations of classification and feature selection algorithms and this much algorithms would require more time to complete if cross-validation was used. Partition was done in a manner that the datasets

included 80% training and 20% testing samples. First the classification algorithms, Naïve Bayes, J48, random forest, k-nearest neighbor, were run solely and then feature selection algorithms were run in combination with classification algorithms. Since the chosen feature selection algorithms, namely Chi-Square, Information Gain and ReliefF are in the type of filter-based approach, they require the predefined number of attributes to be selected. These numbers required by feature selection algorithms were configured as 10, 30 and 50. As a result, the following 40 algorithms were run 10 times for each of the two datasets (one with only permissions and the other with Google Play Store metadata including permissions), comprising totally 80 algorithms.

Table 9- Configurations for classification and feature selection algorithms

| Classifier | Feature Selection Algorithm | # Selected Features |
|---|---|---|
| Naive Bayes | - | - |
| Naive Bayes | Chi-Square | 10 |
| Naive Bayes | Chi-Square | 30 |
| Naive Bayes | Chi-Square | 50 |
| Naive Bayes | Information Gain | 10 |
| Naive Bayes | Information Gain | 30 |
| Naive Bayes | Information Gain | 50 |
| Naive Bayes | ReliefF | 10 |
| Naive Bayes | ReliefF | 30 |
| Naive Bayes | ReliefF | 50 |
| J48 | - | - |
| J48 | Chi-Square | 10 |
| J48 | Chi-Square | 30 |
| J48 | Chi-Square | 50 |
| J48 | Information Gain | 10 |
| J48 | Information Gain | 30 |
| J48 | Information Gain | 50 |
| J48 | ReliefF | 10 |
| J48 | ReliefF | 30 |
| J48 | ReliefF | 50 |
| Random forest | - | - |
| Random forest | Chi-Square | 10 |
| Random forest | Chi-Square | 30 |
| Random forest | Chi-Square | 50 |
| Random forest | Information Gain | 10 |
| Random forest | Information Gain | 30 |
| Random forest | Information Gain | 50 |
| Random forest | ReliefF | 10 |

| Classifier | Feature Selection Algorithm | # Selected Features |
|---|---|---|
| Random forest | ReliefF | 30 |
| Random forest | ReliefF | 50 |
| kNN | - | - |
| kNN | Chi-Square | 10 |
| kNN | Chi-Square | 30 |
| kNN | Chi-Square | 50 |
| kNN | Information Gain | 10 |
| kNN | Information Gain | 30 |
| kNN | Information Gain | 50 |
| kNN | ReliefF | 10 |
| kNN | ReliefF | 30 |
| kNN | ReliefF | 50 |

The full list of results for the algorithms listed at the table above is presented at Appendix E. As it is explained in the parameters used for evaluation of algorithms part, kappa statistic gives information about the agreement of two class values on the classification results. It embodies the information provided by true positive, false positive, true negative and false negative. If the kappa statistic is examined, those parameters are not required to be examined one by one. As a result, evaluations made for all of the algorithms are based on their prediction accuracies and kappa statistics by using paired t-tests. Precision and the area under ROC curve are not selected as the parameters for making comparisons because they exhibit similar behaviors with accuracy of the model. The graphic below displays the values of precision rate, area under ROC curve and accuracy moving together for the proposed dataset in this study (the one including official market metadata with permissions) and the algorithms listed above. Recall rate is also known as true positive rate, so this parameter is not used to evaluate the performance of algorithms in this study.

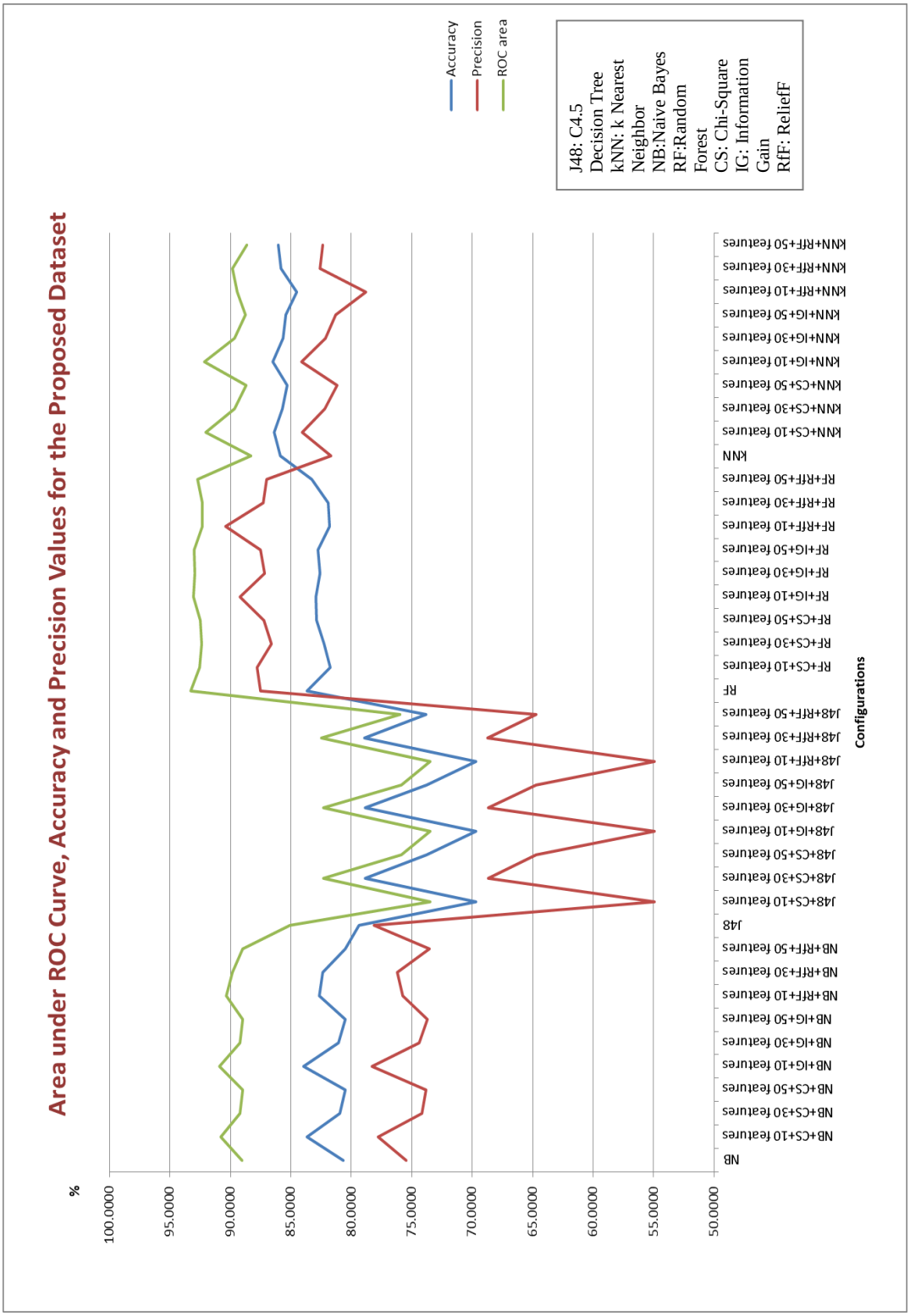



Figure 8- Graph showing the movements of accuracy, precision and area under ROC curve evaluation parameters for the proposed dataset

## 4.2. Evaluation of Official Market Metadata

The main purpose of this study is to answer the question whether Google Play market metadata plays a crucial role while detecting mobile malware and contributes to the detection model with only Android permissions or not. To answer this question, comparisons should be made on two baseline datasets with corresponding classification algorithms. The table at Appendix F presents the comparison of prediction accuracies and kappa statistics for these datasets by displaying the results of paired t-test.

When the results are examined for classification algorithms which are not accompanied by any feature selection algorithm, it can be seen that the prediction accuracy of models improves in half of them as the result of official market metadata addition to the model. Addition of market metadata increases the accuracy from 75.79% to 80.65% for Naïve Bayes, from 83.94% to 85.86% for kNN. The accuracy of the model decreases slightly from 84.81% to 83.66% for random forest when Google Play metadata is added on the permissions. However, the accuracy gets worse a bit more for J48 by decreasing from 83.18% to 79.37%. According to paired t-test results for these values, only the increase of prediction accuracy for Naïve Bayes classifier is statistically significant at 0.05 confidence value. The inspection for the combinations of feature selection algorithms with classification algorithms gives more enlightening information about the predictive accuracy. It is seen that kNN does not improve the accuracy with the addition of market metadata in a statistically significant manner when it is applied without a feature selection algorithm. Yet it produces the statistically more significant accuracies with the addition of market metadata when feature selection algorithms are applied before classification. It gives the higher accuracies for all of the 3 feature selection algorithms, and for all of the 3 levels of selected feature numbers, except ReliefF with 50 features. Naïve Bayes yields more accurate results for proposed detection method in all of 10 combinations. The addition of market metadata decreases the accuracy of the model for all the combinations of J48 and feature selection algorithms but the difference is statistically significant for only the feature selection methods with the number of selected features is 10. Lastly, the random forest does not give quite different accuracies for the two baseline datasets, albeit the dataset including market metadata has higher accuracy than the other one as statistically significant for Information Gain and ReliefF algorithms with 10 selected features.

Kappa statistic shows similar results with prediction accuracy of the model in terms of the effect of official market metadata addition. However there are some different results about kappa statistic. First, the kappa statistic for kNN without any feature selection algorithm points out an improvement by increasing from 0.64 to 0.69 with statistical significance while the difference for prediction accuracy does not have a difference statistically significant. Second, J48 is said above to have significant degradation in terms of accuracy for 3 combinations (Chi-Square, Information Gain and ReliefF algorithms with 10 features) while it has statistically significant decrease in kappa statistic for only 1 combination (ReliefF algorithm with 10 features). Last, though showing statistically significant increase in prediction accuracy for two classification-feature selection algorithm combinations, random forest improves the kappa statistic in a statistically significant manner for only one combination (Information Gain with 10 attributes).

## 4.3. Evaluation of Classification Algorithms

The second research question of this study aims to find the most accurate classification algorithm among Naïve Bayes, k nearest neighbor, random forest and J48 for the task of

Android malware detection with a model comprising of Android permissions and official application market metadata as features. To serve this purpose, paired t-tests were made for the comparison of accuracy values belonging to 4 different classification algorithms on their own, i.e. without any combination with feature selection algorithms. Following tables show the paired t-test results for the comparison of precision values for each classification algorithm with other 3 classification algorithms.

Table 10- Comparison of the prediction accuracy of Naive Bayes classifier with others

| Classifiers[1] / Dataset | Naïve Bayes | J48 | Random Forest | kNN |
|---|---|---|---|---|
| Permissions+ other market metadata | 80.65 | 79.37 | 83.66 v | 85.86 v |

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

Table 11-Comparison of the prediction accuracy of J48 classifier with others

| Classifiers / Dataset | J48 | Naïve Bayes | Random Forest | kNN |
|---|---|---|---|---|
| Permissions+ other market metadata | 79.37 | 80.65 | 83.66 v | 85.86 v |

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

Table 12-Comparison of the prediction accuracy of Random Forest classifier with others

| Classifiers / Dataset | Random Forest | Naïve Bayes | J48 | kNN |
|---|---|---|---|---|
| Permissions+ other market metadata | 83.66 | 80.65* | 79.37 | 85.86 |

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

[1] " v " represents being statistically better than the compared value at 0.05 confidence level and " * " represents being statistically worse than the compared value at 0.05 confidence level

Table 13- Comparison of the prediction accuracy of kNN classifier with others

| Classifiers / Dataset | kNN | Naïve Bayes | J48 | Random Forest |
|---|---|---|---|---|
| Permissions+ other market metadata | 85.86 | 80.65* | 79.37 | 83.66 |

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

When Naïve Bayes is compared with other classification algorithms, it can be realized that the precision accuracy of it is higher than J48 and lower than kNN and random forest algorithms. According to this first comparison, the higher accuracy values of kNN and random forest algorithms have statistically significant difference than Naïve Bayes at 0.05 confidence level. Also it can be said that J48 has a lower accuracy than Naïve Bayes but the difference is not statistically significant, so J48 can be applied too for the detection of Android malware by using market metadata.

The second table compares the accuracy of J48 algorithm with the others and indicates that the higher precision values which are statistically significant belong to random forest and kNN algorithms. The third table presents the comparison of random forest algorithm with the remaining algorithms. The accuracy value of random forest is lower than kNN and higher than Naïve Bayes and J48. This table points out that Naïve Bayes has a lower precision accuracy than random forest algorithm as a statistically significant manner. The last table is for the comparison of the kNN with other algorithms chosen in this study. As it is seen from this table, kNN has the highest accuracy among 4 classification algorithms. Again, the lower precision value of Naïve Bayes has statistically significant difference from kNN.

The highest accuracy value of kNN classification algorithm among others used in this study is obtained when the k parameter (the number of nearest neighbors) is equal to 1. This optimal value of k is applicable for the configuration which the kNN algorithm is used without any feature selection algorithm and is used for the dataset comprising permissions and other official market metadata as feature set. k=1 is selected among the values starting from 1 to 100 by using cross-validation method in a way that classification error is minimized while keeping k value minimum. This result may seem satisfactory in terms of computation time, i.e. the possible shortest computation time obtained in terms of number of neighbors. However using so small k value might result the learned algorithm to be sensitive noise instances in the dataset. Hence, when the evaluation results for the classification algorithms in this study are thought to be utilized, this issue should be considered. (Witten, Frank, & Hall, 2011) claim that the optimum value of k gets smaller when there is less noise in the dataset and the use of cross validation to choose best k value usually gives perfect prediction results. When it is concerned that the focus of this study is not to tune the k parameter for the kNN classification algorithm, the method used to choose k value (also proposed by (Witten, Frank, & Hall, 2011) ) should be accepted as plausible.

To sum up, in this study 4 different classification algorithms are used to evaluate the performance of Android malware detection method which uses permissions and official market metadata as predictor attributes. These supervised learning algorithms are Naïve Bayes, J48,

random forest and kNN. Among these, kNN has the highest accuracy value which is statistically significant than the accuracy value of Naïve Bayes algorithm.

## 4.4. Evaluation of Feature Selection Algorithms

The last goal of the study is to find the most accurate combination of classification algorithms, feature selection algorithms, and the number of selected features. These combinations are listed at the table given in the section 4.1. If the full results listed at Appendix F are examined, it can clearly be seen that the combination having the highest accuracy among all is the kNN classification algorithm which is applied on the dataset after selecting 10 features by using Information Gain feature selection algorithm. This is the answer for the third research question. Additively, each classification algorithm and their combinations with feature selection algorithms are evaluated separately.

Table 14- Accuracy comparison of feature selection algorithms in combination with Naive Bayes

| Combinations / Dataset | NB | NB+CS (10) | NB+ CS (30) | NB+ CS (50) | NB+ IG (10) | NB+ IG (30) | NB+ IG (50) | NB+ RfF (10) | NB+ RfF (30) | NB+ RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 80.65 | 83.68 v | 80.97 | 80.49 | 83.94 v | 81.07 | 80.51 | 82.65 | 82.39 | 80.51 |

v: statistically better than the compared value at 0.05 confidence level
CS: Chi-Square, IG: Information Gain, RfF: ReliefF, NB: Naive Bayes, RF: Random Forest, kNN:k-nearest neighbor

According to the table above, feature selection algorithms generally improve the accuracy of the model when compared to only applying Naïve Bayes, except the ones with the number of features selected are 50. However, there are 2 combinations which are more accurate in a statistically significant manner than the Naïve Bayes applied merely. These combinations are Chi-Square and Information Gain feature selection algorithms with 10 features. In addition to this, as the number of features selected increases, prediction accuracy decreases for these 3 feature selection algorithms. For Chi-Square and Information Gain algorithms; prediction accuracy decreases more when the number of features selected increases from 10 to 30, when compared to the increase of the number of features selected from 30 to 50. This can be seen visually from the graph at Appendix G. ReliefF algorithm behaves differently from this point of view because its discriminative point for the amount of decrease is the shift from 30 features to 50 features. The table below shows the paired t-test result for the comparison of kappa statistics between Naïve Bayes and feature selection algorithms. Kappa statistic displays the similar behaviors with the prediction accuracy of feature selection algorithms in combination with Naïve Bayes. This can be recognized from the graph presented at Appendix G. For kappa statistic, there are 3 combinations which are better in a statistically significant manner than the Naïve Bayes applied merely. As in the accuracy results, Chi-Square and information gain algorithms with 10 features have these higher kappa statistic values. Also ReliefF feature selection algorithm with 10 features yields higher kappa statistic which is statistically significant.

Table 15- Kappa statistic comparison of feature selection algorithms in combination with Naive Bayes

| Combinations / Dataset | NB | NB+ CS (10) | NB+ CS (30) | NB+ CS (50) | NB+ IG (10) | NB+ IG (30) | NB+ IG (50) | NB+ RfF (10) | NB+ RfF (30) | NB+ RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 0.57 | 0.64 v | 0.58 | 0.57 | 0.65 v | 0.59 | 0.57 | 0.62 v | 0.62 | 0.57 |

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

For J48 classification algorithm, applied feature selection algorithms decrease accuracy values for all combinations. The decrease of accuracy is statistically significant for Chi-Square and information gain algorithms with 10 features. Hence, it may not be a good choice to apply one of these feature selection algorithms if the prediction accuracy is considered. In addition to this, all 3 feature selection algorithms produce very close accuracy values. The produced accuracies are even equal for Chi-Square and information gain algorithms for the same number of selected features. This is probably caused by the rationale of the decision tree. Independent from an additional feature selection algorithm, decision tree applies feature selection by using information gain parameter inside; it chooses the best attributes for classification. The table for kappa statistic comparison of feature selection algorithms in combination with J48 also points out that 3 feature selection algorithms give the same results when the numbers of selected features are the same. However an important indication may be that the use of 30 features for those feature selection algorithms serves the purpose of obtaining more balanced results (higher kappa statistic value) than the results of pure J48 classification algorithm.

Table 16- Accuracy comparison of feature selection algorithms in combination with J48

| Combinations / Dataset | J48 | J48+CS (10) | J48+ CS (30) | J48+ CS (50) | J48+ IG (10) | J48+ IG (30) | J48+ IG (50) | J48+ RfF (10) | J48+ RfF (30) | J48+ RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 79.37 | 69.69 * | 78.85 | 73.77 | 69.69 * | 78.85 | 73.77 | 69.69 * | 78.91 | 73.81 |

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

Table 17-Kappa statistic comparison of feature selection algorithms in combination with J48

| Combinations / Dataset | J48 | J48 +CS (10) | J48+ CS (30) | J48+ CS (50) | J48+ IG (10) | J48+ IG (30) | J48+ IG (50) | J48+ RfF (10) | J48+ RfF (30) | J48+ RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 0.56 | 0.42 | 0.58 | 0.47 | 0.42 | 0.58 | 0.47 | 0.42 | 0.58 | 0.47 |

When the tables of accuracy comparison for feature selection algorithms in combination with kNN and random forest are examined, it is seen that none of the combinations has a statistically significant difference than the mere classification algorithms. Additively, for random forest algorithm, Chi-Square feature selection algorithm displays a regular increase when the number of features increases from 10 to 50. However information gain algorithm does not present a regular decrease or increase in prediction accuracy. Instead, it first decreases the accuracy from 10 features to 30 features and then increases from 30 to 50, but the differences are slight. Hence for this study, the number of features selected for information gain feature selection algorithm when applied with J48 does not matter from the accuracy point of view. In addition, the accuracy of the J48algorithm in combination with ReliefF does not change too much when the number of features selected is changed from 10 to 30. Yet the increase of the accuracy is more apparent as the number of selected features increases from 30 to 50. For kNN, it can be claimed that Chi-Square and information gain algorithms display the similar behaviors according to the number of selected features and have close accuracy values. However, ReliefF algorithm when applied together with kNN classification algorithm shows an increasing trend for accuracy values in conjunction with increasing number of features. To visually clarify these conclusions, graphs can be applied at Appendix G.

Table 18- Accuracy comparison of feature selection algorithms in combination with Random Forest

| Combinations / Dataset | RF | RF+CS (10) | RF +CS (30) | RF +CS (50) | RF + IG (10) | RF + IG (30) | RF + IG (50) | RF + RfF (10) | RF + RfF (30) | RF + RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 83.66 | 81.74 | 82.28 | 82.85 | 82.93 | 82.59 | 82.75 | 81.8 | 81.92 | 83.26 |

Table 19- Accuracy comparison of feature selection algorithms in combination with kNN

| Combinations / Dataset | kNN | kNN +CS (10) | kNN +CS (30) | kNN +CS (50) | kNN + IG (10) | kNN + IG (30) | kNN + IG (50) | kNN + RfF (10) | kNN + RfF (30) | kNN + RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 85.86 | 86.41 | 85.7 | 85.32 | 86.47 | 85.66 | 85.4 | 84.49 | 85.82 | 86.03 |

Lastly, kappa statistics of feature selection algorithms in combination with random forest and kNN classification algorithms are compared at the following two tables. None of the combinations has higher kappa statistic value in a statistically significant manner than the pure kNN and random forest algorithms. Combinations have very close kappa statistic values for both of random forest and kNN classification algorithms. For kNN, selecting number of features as 30 or 50 does not differ because they yield equal kappa statistic values for all 3 feature selection algorithms.

Table 20-Kappa statistic comparison of feature selection algorithms in combination with Random Forest

| Combinations / Dataset | RF | RF+CS (10) | RF +CS (30) | RF +CS (50) | RF + IG (10) | RF + IG (30) | RF + IG (50) | RF + RfF (10) | RF + RfF (30) | RF + RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 0.62 | 0.57 | 0.59 | 0.6 | 0.6 | 0.59 | 0.6 | 0.57 | 0.58 | 0.61 |

Table 21-Kappa statistic comparison of feature selection algorithms in combination with kNN

| Combinations / Dataset | kNN | kNN +CS (10) | kNN +CS (30) | kNN +CS (50) | kNN + IG (10) | kNN + IG (30) | kNN + IG (50) | kNN + RfF (10) | kNN + RfF (30) | kNN + RfF (50) |
|---|---|---|---|---|---|---|---|---|---|---|
| Permissions+ other market metadata | 0.69 | 0.7 | 0.68 | 0.68 | 0.7 | 0.68 | 0.68 | 0.66 | 0.69 | 0.69 |

## 4.5. Summary of Findings

The analysis studies conducted aimed to answer aforementioned 3 research questions. For the first research question it has been found that the addition of official market metadata on Android permissions as predictor variables improves the accuracy of the model and the agreement of class values on the results for Naïve Bayes and kNN classification algorithms. According to the results of random forest algorithm, addition of market data does not show the signs of significant improvement. J48 is affected negatively with the addition of market metadata but this effect was founded to be statistically insignificant. Two different datasets one of which consisting of only Android permissions and the other one adding market metadata as predictor attributes were used for the comparisons made to answer first research question. The second research question investigates the most accurate classification algorithm among Naïve Bayes, J48, random forest and kNN for the problem of Android malware detection using official market metadata. To answer this question one baseline dataset which includes official market metadata beside permissions is used. According to results, kNN has the best performance in terms of prediction accuracy and its accuracy value is higher than the accuracy of Naïve Bayes algorithm as statistically significant. The third and the last research question of this study is about finding the most accurate combination of classification, feature selection algorithms and the number of features selected for feature selection algorithms. Comparisons were made on the dataset which comprise of official market metadata and requested permissions as feature set. kNN classification algorithm applied together with Information Gain feature selection method with 10 features yields the highest accuracy among 36 combinations. Also it is found that applying a feature selection method does not improve the performance of classification algorithms always. Also, selecting more or less features does not have an improving effect on prediction results always.

# CHAPTER 5

# DISCUSSION AND CONCLUSION

This chapter concludes the study by summarizing the adopted detection method, feature selection and supervised classification algorithms used in this study, and the results obtained from them are discussed. Also the contributions of the study are given and the limitations to which the study is exposed are indicated. In addition, the future works are proposed to be fulfilled which have not been handled under the scope of this study but may improve the generalizability of the results for this study.

## 5.1. Discussion and Conclusion

There are a lot of methodologies developed by researchers to tackle with the mobile malware detection problem. Especially Android devices are examined closer due to their open structures and proneness to malicious applications because of the reasons discussed in section 2.1. To carry out the mobile malware detection task, dynamic and static analysis methods are applied mainly. Dynamic analysis method is not preferred in this study because of its high cost in deployment environment, and requirement of complicated skills and manual investigating. ( Zhu & Peiravian, 2013) ( Wu , Mao, Wei, Lee, & Wu, 2012) Instead, a feature-based automated static analysis method is preferred. To automate detection task, supervised machine learning algorithms are applied by using permissions requested from the user at application install time and official market metadata as predictor variables. 4 different classification algorithms, Naïve Bayes, k-nearest neighbor, J48 (java implementation of C4.5 decision tree) and random forest (a kind of ensemble method) were chosen to be trained and make predictions on the class values of unknown applications. Type of classification in this study is binary classification and the class attribute has "benign" and "malicious" values to label applications. Performances of algorithms are evaluated by using prediction accuracy and kappa statistic.

Before carrying out the main analyses, a pilot study was performed in order to see the feasibility of the model. In this pilot study, two baseline datasets were used to evaluate the contribution of

official market metadata on permissions. One of them included only permissions as feature set and the other one comprised of permissions together with official market metadata. To construct these datasets, Google's official application market and VirusTotal free online malware scanner are utilized. Features to be used in classification algorithms, like permissions, developer names, type of applications, download number of application etc., were collected from Google Play by using a web crawler and a java application. Collected applications were labeled as malicious or benign by scanning them at VirusTotal website. For the pilot study, applications were labelled as malicious if they have been identified by 2 or more AV engines as malicious. The findings of this pre-assessment showed that the addition of official market metadata to feature set improved the accuracy and reduced the false alarms. The improvement of the detection results was somehow acceptable but still the prediction accuracy and the false positive rate were not satisfactory. Just as it would be done for a part of main study, feature selection methods applied at this step. However, the application of feature selection algorithms for selecting the attributes which would be used as input by classification algorithms did not produce significantly better results.

To overcome the high false positive rates and to improve the prediction accuracy two options were evaluated. First, the number of benign and malicious applications in the dataset was balanced and by reducing the number of benign applications. Balancing the total number of benign and malicious applications did not improve the detection results, but balancing them under each application category decreased the false alarms. As the second option, labelling method of applications as benign or malicious has been revised. Applications which have not been identified as malicious by any of the antivirus engines were labeled as benign. To label malicious applications, an experiment was conducted and the number of AV engines identifying an application as malware was used as decision criteria. The detection count of malicious applications was increased from 2 to 10 one by one and 9 different datasets were constructed. While constructing these datasets, the ambiguous applications were omitted from the dataset. For instance, to construct the dataset which includes malwares identified by 5 or more AV engines, applications were labeled as benign if their detection count are zero and the remaining ones which have detection count ranging from 1 to 4 were eliminated. In this experiment, the prediction results of Naïve Bayes were used to select proper malware detection count. The reason lying behind to choose Naïve Bayes for pilot studies and experiment is its low computation time allowing a fast pre-assessment. The results of the experiments pointed out the selection of the detection count as 8 to label malicious applications. Finally on this dataset balancing operation was applied to reduce the false alarms by equalizing the number of malicious and benign applications under each application category.

By using final baseline dataset, analyses were made to answer 3 research questions. The main purpose of this study is to examine whether the contribution of official market metadata on requested permissions is meaningful to classify applications. Previous studies has utilized permissions, API calls, intent messages, and main components (content providers, broadcast receivers, activities, and services) of Android applications ( Enck, Ongtang, & McDani, 2009) ( Wu , Mao, Wei, Lee, & Wu, 2012) ( Zhu & Peiravian, 2013) for statically analyzing them. (Glodek & Harang, 2013) add the combinations of frequently used permissions and native code as features for detecting Android malwares. Considering those studies and the claims that Android permissions are not sufficient on their own to explain the malicious behaviors ( Meurer & Wismüller, 2012) ( Enck, Ongtang, & McDani, 2009), official market metadata is proposed as additional feature to the requested permissions in this study. (Abu Samra, Yim, & Ghanem, 2013) and (Bose, Hu, Shin, & Park, 2011) have also used official market metadata as static

features, but they applied unsupervised clustering techniques to categorize applications. Differently, in this study permissions and official application market metadata were used as features for training supervised classification algorithms.

In order to answer this first question, two baseline datasets were compared by applying classification and feature selection algorithms. One of the datasets was the one obtained as final baseline dataset and comprised of permissions together with official market metadata. The other one was constructed from the final dataset in a way that it included only permissions as feature set. When the results of classification algorithms and their combinations with feature selection algorithms were examined, it has seen that the addition of market metadata increased the accuracy of the model for Naïve Bayes and kNN classification algorithms. Their combinations with feature selection algorithms also signed to increase of prediction accuracy and kappa statistic. Obtaining higher kappa statistics means the proposed features provides a model with less false alarms and more agreement between class values on classification results. Additionally, random forest algorithm for the proposed model did not produce significantly different classification results than the model with only permissions. It increased the prediction accuracy of the model as statistically significant for some combinations (of classification and feature selection algorithms) and decreased slightly for the remaining ones. However J48 algorithm was affected negatively by the addition of market metadata to the model. Most of the combinations of feature selection algorithms with J48 showed a decrease of approximately 8% in model accuracy. Yet according to paired t-test results, only 3 combinations worsened the accuracy and 1 combination worsened the kappa statistic in a statistically significant way.

The second research question was about finding the most accurate classification algorithm applied on the proposed dataset in this study. To answer this question, only the accuracies of classification algorithms were compared without using their combinations with feature selection algorithms. Results showed that the kNN outperformed the other 3 classification algorithms with 85.86% accuracy value.

To answer the last research question, the effect of 3 feature selection algorithms on the classification task of applications by using permissions and official market metadata was investigated. As being type of filtered feature selection methods, Chi-Square, Information gain and ReliefF algorithms were chosen. Since filtered feature selection methods require the number of features to be selected from the implementer, different numbers were identified in order to avoid the use of too many or few features. The identified numbers for the features to be selected by feature selection algorithms were 10, 30 and 50. The third research question investigates the most accurate combination of classification and feature selection algorithms, and the identified number of features to be selected. According to the results, kNN yields the highest accuracy when it is combined with Information Gain feature selection algorithm and 10 features are selected.

Lastly, the results of this study and the other studies mentioned in the literature review part which apply automated feature-based machine learning approach as static Android malware detection method are presented at the table below. Each study applies different machine learning algorithms on different datasets. Hence, it would not be plausible to compare them exactly in terms of their prediction performance measures. However the results of the studies obtained by using the datasets, feature sets, and the machine learning algorithms presented on the table below would guide the researchers who intend to develop or apply a feature-based static Android malware detection method. When the performance measures of this study are

compared the others at the table below, the accuracy and TPR are acceptable. However, the FPR rate needs to be improved more to avoid false alarms. This drawback of the proposed model may be caused by the method used to label applications as malicious or benign. In this study, both the malicious and benign applications are labeled by querying the analysis results of them on VirusTotal, but this is not a verified method by previous studies. The remaining studies choose to study with the malicious applications proven before by other studies.

Table 22- Summary of the methods and the findings of related studies to the proposed method in this study

| Title of the Study | Dataset Used | Machine Learning Algorithms Used | Features Used in Selected Classification Algorithms | Results |
|---|---|---|---|---|
| Droidmat: Android Malware Detection through Manifest and API Call Tracing | Includes 238 Android malware collected from a public Android dataset, Contagio mobile, and 1500 benign applications downloaded from official Android market and verified through the website of VirusTotal malware detection community . | kNN (with k=1) and Naive Bayes classification algorithms, EM and K means clustering algorithms (SVM to decide number of clusters) | Requested permissions, intent messages passing, API calls and components of applications(activity, service and broadcast receiver) | The combination of k means and kNN was chosen as having the highest recall rate=0.87 and precision=0.97 |
| Machine Learning for Android Malware Detection Using Permission and API Calls | 610 malware from Malware Genome Project[2] (they eliminated the repetitive samples among 1250 applications) and 1250 benign applications downloaded from Google Play (top 50 free applications under 25 application categories) | SVM, Decision tree and Bagging classification algorithms | Permissions and API calls | They claim that the best classification algorithm is Bagging in terms of detection rate (TPR). It has 96.39% accuracy, 94.9% precision 94.1% recall rate with the regarded features and the dataset. |

[2] Y. Zhou and X. Jiang, Dissecting android malware: Characterization and evolution Security and Privacy (SP), 2012 IEEE Symposium on Security and Privacy.

| Title of the Study | Dataset Used | Machine Learning Algorithms Used | Features Used in Selected Classification Algorithms | Results |
|---|---|---|---|---|
| Rapid Permissions-based Detection and Analysis of Mobile Malware Using Random Decision Forests | Malicious applications from Malware Genome Project (random 500 of them were chosen) and 500 benign applications downloaded from 3rd party markets. Contagio mobile website (16 malwares) for testing the capability of model for novel malware. | Random forest classification algorithm | Requested permissions, broadcast receiver, presence of embedded Android applications, native code | TPR: 92%<br>FPR: 3% |
| Proposed method in this study | 2528 instances collected from the Android's official application market, GooglePlay, and labeled via VirusTotal (907 malicious and 1621 benign applications) | Naïve Bayes, J48, kNN, and random forest classification algorithms, Chi-Square, Information Gain and ReliefF feature selection algorithms | Requested permissions and other metadata of applications presented on the official market (like app category, developer name, download range, size of application) | The best combination kNN + Information Gain (with 10 features). It has 86.47% accuracy, 91.86% TPR, 23.17% FPR, 0.70 kappa statistic |

To sum up, in this study mainly the contribution of official market metadata on explaining the malicious behaviors of Android applications was investigated. The addition of market metadata improved the accuracy and false alarms of the model which comprise of only Android permissions for half of the selected classification algorithms (Naïve Bayes and kNN). In addition to this, it was shown that a detection method can be built by using a free public data provided by Google's official application market and VirusTotal free online AV engine. The other contribution of this study is to include all the free applications presented on the Google Play without limiting to some application categories.

## 5.2. Limitations and Further Research

The dataset used to train and evaluate the performance of supervised machine learning algorithms in this study contains the top free applications presented by Google's official application market. The paid applications might be used for analysis but the crawler used in this study allows downloading only free applications, like other tools used for downloading Android applications from official market. The official market is also open to malevolent developers who download legitimate applications from the market and upload the repacked ones after

injecting malicious code into them. Since there is no reason for them not to apply this on their paid applications, inspecting paid applications may make sense. On the other hand, hackers also can be thought not to prefer paid applications to download and make them malicious for the sake of cost and simplicity. Hence, for an application having a paid version may be sign of being more trustable. At least, adding a feature to the dataset which indicates that an application has a paid version or not may help to explain malicious behaviors of applications better. In addition, if the Google Play presented the hash values of applications, without the need for downloading and calculating hash values, paid applications would also be inspected by querying them on VirusTotal.

Another limitation of the study is the use of Google's official application market for mobile malware detection task. The dataset can be extended in a way that it includes the applications from other third-party application markets and/or websites. In spite of the fact that users are required to root their smartphones to download Android applications from unofficial markets and websites, they may prefer this for enhancing their phones' capabilities.

The results of the study can be evaluated as acceptable in terms of accuracy, but the unexplained part of the malicious behaviors by the proposed model is caused by some limitations. First, the downloaded applications have not been reengineered to extract the part of the apk files. Therefore, vital part of an apk, Android Manifest.xml, could not be examined to find out the intent messages and application components (activities, services, content providers, and broadcast receivers). Also the bytecode of an application could not be investigated to explore the API calls. The reason for not choosing to decompress applications is that all the applications presented on official market do not let to be reverse-engineered. This may be caused by some applications' being commercial products or brands of companies or being protected from piracy by their developers and issued to copyrights. Second, due to the nature of static analysis, some hidden capabilities of malwares could not be revealed. For instance, as it is stated by ( Wu , Mao, Wei, Lee, & Wu, 2012) some Android malwares are capable of downloading the actual payloads from remote sites by using the internet connection, so static detection methods are not proficient enough in this regard.

As feature work, different classification and feature selection algorithms can be applied for Android malware detection method proposed in this study. For example, ( Zhu & Peiravian, 2013) argue that the use of ensemble learning methods may lead to better results for imbalanced datasets because of their structure based on consensus. (Jamali, Bazmara, & Jafa, 2012) claim compliance of some feature selection methods with specific classification algorithms. Therefore, the accordance of feature selection and classification algorithms or other factors may be regarded for the selection of those algorithms in the following studies. Additively, a longitudinal study may be considered because it may strengthen the findings of the study. In time, the number of AV engines analyzing the applications or the detection rate of malicious applications can be changed, and also the ones found to be benign before can be labeled as malicious. However, the loss of data should be considered in longitudinal study, because the analysis results of some applications for some AV engines might be removed after a while owing to time out. Also, some malicious applications with high detection number by AV engines have become clean applications in time according to the analysis results on VirusTotal. The reasons underlying this are not known precisely, but guessed as the clean of legitimate applications from malicious codes if they have been injected into before and reload of them to the market.

# REFERENCES


Abela, K. J., Angeles, D. K., Delas Alas, J. R., Tolentino, R. J., & Gomez, M. A. (2013). The Society of Digital Information and Wireless Communications. An Automated Malware Detection System for Android using Behavior-based Analysis (s. 1-11).

Abu Samra, A., Yim, K., & Ghanem, O. (2013). 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. Analysis of Clustering Technique in Android Malware Detection (s. 729-733). IEEE.

Al-Ibrahim, A. (2011). Discretization of Continuous Attributes in Supervised Learning. The Research Bulletin of Jordan ACM, s.158-166.

Ali,J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random Forests and Decision Trees. IJCSI International Journal of Computer Science Issues.

APK Downloader. Retreived July, 2014, from APK Downloader: http://apps.evozi.com/apk-downloader/

Bai,J., Wang, J., & Zou, G. (2014). A Malware Detection Scheme Based on Mining Format Information. The Scientific World Journal.

Benyamin, D. (2012, November 10). A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System. Retrieved July, 2014, from http://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

Bhatia, N., & Vandana. (2010). (IJCSIS) International Journal of Computer Science and Information Security. Survey of Nearest Neighbor Techniques.

Blue Coat Systems. (2014). 2014 Mobile Malware Report. 20(1), 37-46.

Blue Coat Sytems. (2013, July 18). 2013 Mobile Malware Report. How Users Drive the Mobile Threat Landscape.

Bose, A., Hu, X., Shin, K. G., & Park, T. (2008). MobiSys '08 Proceedings of the 6th international conference on Mobile systems, applications, and services. Behavioral Detection of Malware on Mobile Handsets. Colorado, USA: ACM.

Bose, A., Hu, X., Shin, K. G., & Park, T. (2011). Security and Privacy in Mobile Information and Communication Systems Third International ICST Conference. Android Market Analysis with Activation Patterns (s. 1-12). Aalborg, Denmark: Springer.

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

Cios,K., Pedrycz, W., Swiniarski, R., & Kurgan, L. (2007). Unsupervised Discretization Algorithms. Data Mining: A Knowledge Discovery Approach (s. 237). in Springer.

Cunningham, P., & Delany, S. J. (2007). k-Nearest Neighbour Classifiers. UCD School of Computer Science and Informatics.

Damopoulos, D., Kambourakis, G., & Portokalidis, G. (2014). EuroSec '14 Proceedings of the Seventh European Workshop on System . The Best of Both Worlds. A Framework for the Synergistic Operation of Host and Cloud Anomaly-based IDS for Smartphon

Dash,R., Paramguru, R., & Dash, R. (2011). Comparative Analysis of Supervised and Unsupervised Discretization Techniques. International Journal of Advances in Science and Technology.

Demiröz, A. (2013, June 11). Akdeniz/google-play-crawler. Retrieved July, 2014, from https://github.com/Akdeniz/google-play-crawler/blob/master/README.md: https://github.com/Akdeniz/google-play-crawler/blob/master/README.md

developer.android.com. (2014, Aug). API Guides. Retreived Aug, 2014, from Android Developers.

Dini, G., Martinelli, F., Saracino, A., & Sgandurra, D. (2012). MMM-ACNS'12 Proceedings of the 6th international conference on Mathematical Methods, Models and Architectures for Computer Network Security. MADAM: a multi-level anomaly detector for android malware

Elkan, C. (1997). Boosting and Naive Bayesian Learning. San Diego.

Enck,W., Ongtang, M., & McDani, P. (2009). CCS '09 Proceedings of the 16th ACM conference on Computer and communications security. On lightweight mobile phone application certification (s. 235-245). Chicago, Illinois, USA: ACM.

Geng,X., Liu, T.-Y., Qin, T., & Li, H. (2007). SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. Feature selection for ranking (s. 407-414). Amsterdam, The Netherlands

Glodek, W., & Harang, R. (2013). Military Communications Conference, MILCOM 2013 - 2013 IEEE . rapid permission based detection and analysis of mobile malware using random decision forests (s. 980-985). San Diego, CA : IEEE.

Google. (2014, Aug). Android Developer Help. Retreived Aug, 2014, from Google Support: https://support.google.com/googleplay/android-developer/answer/1295940?hl=en

Google. (2014, Aug). Android Developer Help. Retreived 2014, Aug, from Google Support: https://support.google.com/googleplay/android-developer/answer/188189?hl=en&ref_topic=345098

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. (2009). SIGKDD Explorations. The WEKA Data Mining Software: An Update.

Horning, N. (2010). International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences 2010. Random Forests:An Algorithm For Image Classification And Generation Ofcontinuous Data Sets. Hanoi, Vietnam.

HP. (2013, May). Designing a Defense for Mobile Applications Examining an ecosystem of risk.

Imam,I., Michalski, R., & Kerschberg, L. (1993). In Proceeding of the First International Workshop on Knowledge Discovery in Databases. Discovering attribute dependence in databases. Washington D.C.

iMacros. Introducing iMacros. Retreived July, 2014, from iMacros wiki.

Jamali, I., Bazmara, M., & Jafa, S. (2012). Feature Selection in Imbalance data sets. IJCSI International Journal of Computer Science Issues.

Joita, D. (2010). Unsupervised Static Discretization Methods In Data Mining. MegaByte.

Kira,K., & Rendell, L. A. (1992). ML92 Proceedings of the ninth international workshop on Machine learning. A practical approach to feature selection (s. 249-256). Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. F. R. Bergadano in, Machine Learning: ECML-94 (s. 171-182). Springer-Verlag.

Koshal, J., & Bag, M. (2012). Cascading of C4.5 Decision Tree and Support Vector Machine for Rule Based Intrusion Detection System. International Journal of Computer Network and Information Security, 8-20.

Kotsiantis, S., & Kanellopoulos, D. (2006). Discretization Techniques: A recent survey. GESTS International Transactions on Computer Science and Engineering, 47-58.

La Polla, M., Martinelli, F., & Sgandurra, D. (2013, March 15). A Survey on Security for Mobile Devices. IEEE.

Lawton, G. (2008, May). Is It Finally Time to Worry about Mobile Malware? IEEE .

Meurer, S., & Wismüller, R. (2012). Security and Privacy in Mobile Information and Communication Systems- 4th International Conference, MobiSec 2012. APEFS: An Infrastructure for Permission-Based Filtering of Android Apps (s. 1-11). Frankfurt am Main, Germany

Microsoft. (2014, Aug). Ensuring Data Integrity with Hash Codes. Retreived Aug, 2014, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/f9ax34y5%28v=vs.110%29.aspx

Mitchell, M. (2010, January 11). MachineLearningWinter2010/pdfslides. Retreived July, 2014, from CS 445/545:Computer Science Departments Winter Quarter 2010: http://web.cecs.pdx.edu/~mm/MachineLearningWinter2010/pdfslides/EvaluatingHypotheses.pdf

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill Science/Engineering/Math.

Mitov, I., Ivanova, K., Markov, K., Velychko, V., Stanchev, P., & Vanhoof, K. (2008). Comparison of Discretization Methods for Preprocessing Data. International Book Series "Information Science and Computing" (s. 31-39).

Novakovic, J., Strbac, P., & Bulatovic, D. (2011). Toward Optimal Feature Selection Using Ranking Methods and Classification Algorithms. Yugoslav Journal of Operations Research, 119-135.

Orthacker, C., Teufl, P., Kraxberger, S., Lackner, G., Gissing, M., Marsalek, A., et al. (2011). Security and Privacy in Mobile Information and Communication Systems-Third International ICST Conference, MobiSec 2011. Android Security Permissions

Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M., & Ioannidis, S. (2014). EuroSec '14 Proceedings of the Seventh European Workshop on System Security. Rage against the virtual machine: hindering dynamic analysis of Android malware

Porter Felt, A., Finifter, M., Chin, E., Hanna, S., & Wagner, D. (2011). SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. A Survey of Mobile Malware in the Wild (pp. 3-14). New York: ACM.

QIMacros. How to Determine Histogram Bin Intervals. Retreived July, 2014, from QIMacros: http://www.qimacros.com/quality-tools/how-to-determine-histogram-bin-interval/

Quinlan, J. (1986). Induction of Decision Trees. Machine Learning 1, 81-106.

Quinlan, J. (1993). C4.5: Programs for Machine Learning. San Mateo: CA: Morgan Kaufmann.

Rapit7. (2013, July 25). Mobile Security Guide: Protect Your Organization from Mobile Malware .

Schreckling, D., Huber, S., Höhne, F., & Posegga, J. (2013). Information Security Theory and Practice. Security of Mobile and Cyber-Physical Systems-7th IFIP WG 11.2 International Workshop, WISTP 2013. URANOS: User-Guided Rewriting for Plugin-Enabled ANdroid ApplicatiOn Security

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). "Andromaly": a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems, 161-190.

Shabtai, A., Fledel, Y., & Elovici, Y. (2010). 2010 International Conference on Computational Intelligence and Security. Automated Static Code Analysis for Classifying Android Applications Using Machine Learning (s. 329-333). Nanning: IEEE.

Shmueli, G., Patel, N. R., & Bruce, P. C. (2010). Data Mining For Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner. John Wiley & Sons.

Sikonja, M. R., & Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning Journal, 23-69.

Suarez-Tangil , G., Tapiador, J., Peris-Lopez, P., & Blasco, J. (2014). Dendroid: A text mining approach to analyzing and classifying code structures in Android malware families. Elsevier, 1104-1117.

Sybase. (2011, April 11). It's All About Security: Things To Know Before You Open The Doors To Smartphones And Tablets In Your Enterprise.

Symantec. (2011, June 28). A Window into Mobile Device Security . Examining the security approaches employed in Apple's iOS and Google's Android.

Symantec. (2012, October). Securing the Mobile App Market. How Code Signing Can Bolster Security for Mobile Applications.

Thabtah, F., Eljinini, M., Zamzeer, M., & Hadi, W. (2009). Naïve Bayesian Based on Chi Square to Categorize Arabic Data. Communications of the IBIMA.

USA Department of Commerce. (2012, July). Guidelines for Managing and Securing Mobile Devices in the Enterprise.

Uysal, A. K., & Gunal, S. (2012). A novel probabilistic feature selection method for text classification. Elsevier Science Publishers, 226-235.

VirusTotal. Frequently Asked Questions. Retrieved July, 2014, from VirusTotal: https://www.virustotal.com/en/faq/#shortcuts

VirusTotal. VirusTotal. Retreived July, 2014, from VirusTotal: https://www.virustotal.com/

Vryniotis, V. (2014, January 20). Using Feature Selection Methods in Text Classification. Retreived July, 2014, from Machine Learning Blog&Software Development News: http://blog.datumbox.com/using-feature-selection-methods-in-text-classification/

weka.(2009, March 17). How weka deal with categorical data . Retreived July, 2014, from weka: http://weka.8497.n7.nabble.com/How-weka-deal-with-categorical-data-td3144.html

Wikipedia. (2014, July 12). Confusion Matrix. Retreived July, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Confusion_matrix

Wikipedia. (2014, July 8). Histogram. Retreived July, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Histogram

wikipedia. (2014, July 23). Linux kernel. Retreived July, 2014, from wikipedia: http://en.wikipedia.org/wiki/Linux_kernel

Wikipedia. (2014, July 19). Wikipedia. Retreived July, 2014, from MD5: http://en.wikipedia.org/wiki/MD5

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques Third Edition. Burlington: Morgan Kaufmann.

Wu ,D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., & Wu, K.-P. (2012). 2012 Seventh Asia Joint Conference on Information Security. DroidMat: Android Malware Detection through Manifest and API Calls Tracing (s. 62-69). Tokyo: IEEE.

Yang,J., & Li, Y.-P. (2006). International Conference on Intelligent Computing, ICIC 2006. Orthogonal Relief Algorithm for Feature Selection (s. 227-234). Kunming, China: Springer.

Yang,Y., & Pedersen, J. (1997). Yang, Y., & Pedersen, J. O. (1997, July). A comparative study on feature selection in text categorization. ICML, (s. 412-420).

Yu, L., & Liu, H. (2003). In Proceedings of ICML 2003. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution, (s. 856-863). Washington D.C.

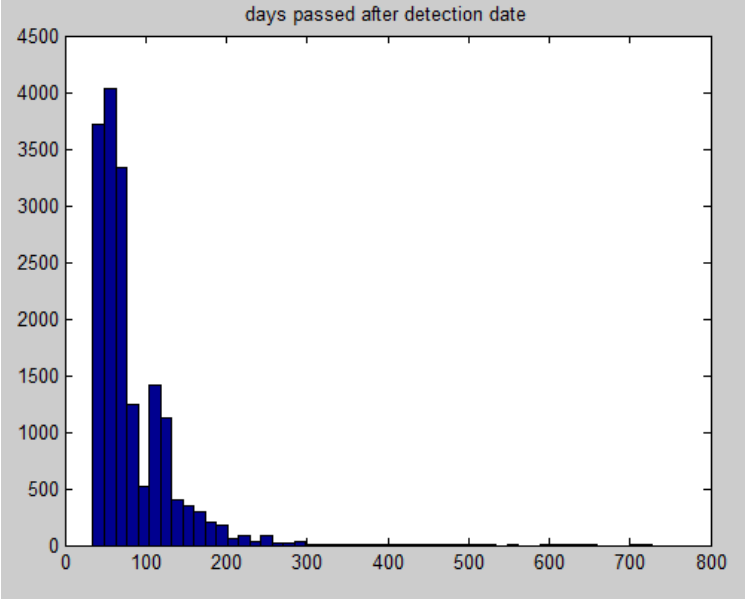
Zhu, X., & Peiravian, N. (2013). 2013 IEEE 25th International Conference on Tools with Artificial Intelligence. Machine Learning for Android Malware Detection Using Permission and API Calls (s. 300-305). Herndon, VA: IEEE.

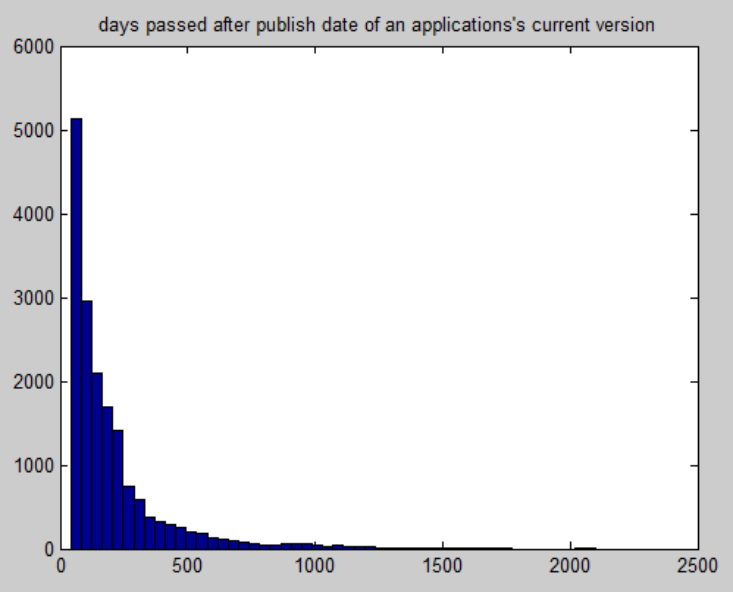
# APPENDICES

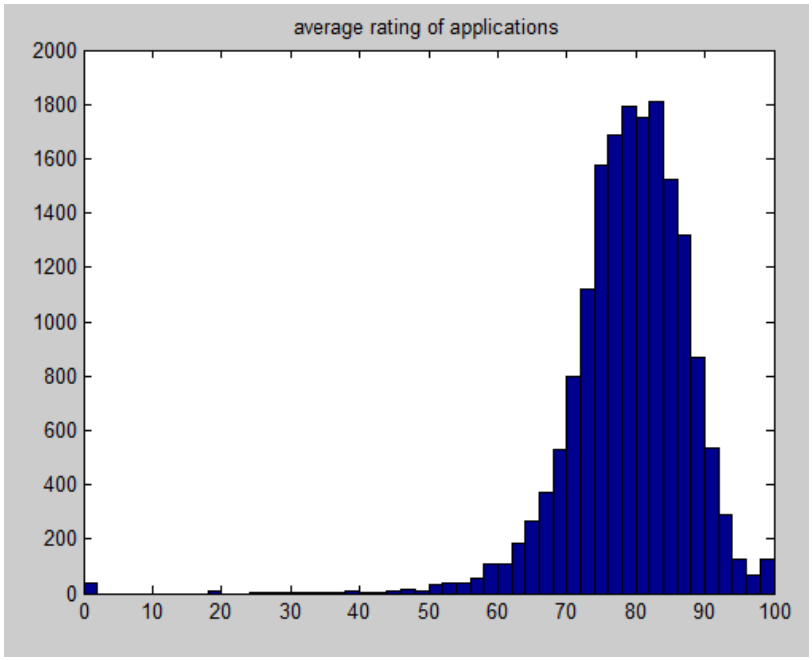
## Appendix A: List of Application Categories and Application Download Ranges

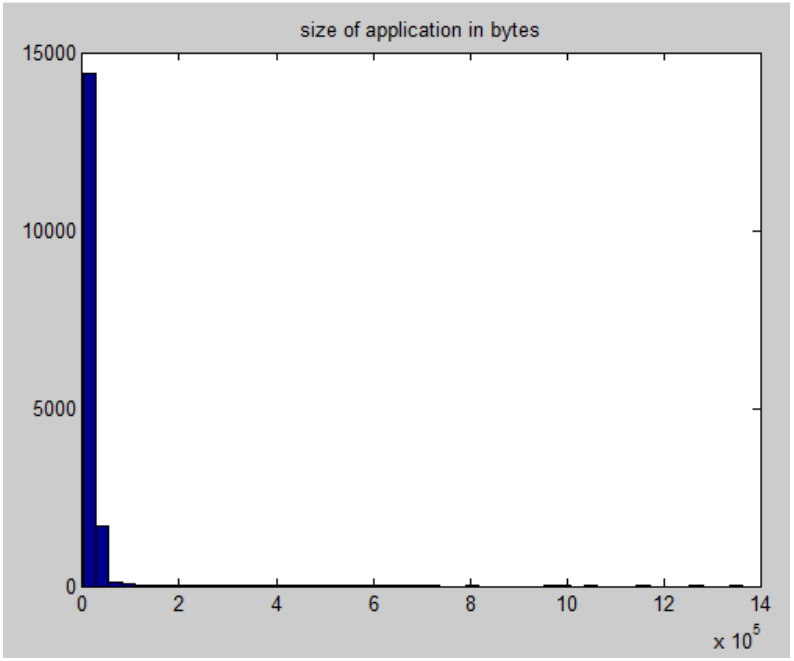
| APPLICATION CATEGORIES | | DOWNLOAD RANGES |
|---|---|---|
| Application Type | Game Type | 1-5 |
| Books and Reference | Action | 5-10 |
| Books and Reference | Adventure | 10-50 |
| Business | Arcade | 50-100 |
| Comics | Board | 100-500 |
| Communication | Card | 500-1,000 |
| Education | Casino | 1,000-5,000 |
| Entertainment | Casual | 5,000-10,000 |
| Finance | Educational | 10,000-50,000 |
| Health and Fitness | Family | 50,000-100,000 |
| Libraries and Demo | Music | 100,000-500,000 |
| Lifestyle | Puzzle | 500,000-1,000,000 |
| Media and Video | Racing | 1,000,000-5,000,000 |
| Medical | Role Playing | 5,000,000-10,000,000 |
| Music and Audio | Simulation | 10,000,000-50,000,000 |
| News and Magazines | Sports | 50,000,000-100,000,000 |
| Personalization | Strategy | 100,000,000-500,000,000 |
| Photography | Trivia | 500,000,000-1,000,000,000 |
| Productivity | Wallpaper | 1,000,000,000-5,000,000,000 |
| Shopping | Word | |
| Social | | |
| Sports | | |
| Tools | | |
| Transportation | | |
| Travel | | |
| Wallpaper | | |
| Weather | | |

## Appendix B: Histograms of Continuous Attributes




average rating of applications
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
80
90
100


size of application in bytes
15000
10000
5000
0
0
2
4
6
8
10
12
14
x 10^5

## Appendix C: Paired t-test Results for Comparison of Predictive Accuracies and Kappa Statistics to Choose Baseline Datasets

Table 23- Paired t-test results for accuracy comparison to choose baseline datasets

| Datasets / Classifier | Detection Count=3 | Detection Count=4 | Detection Count=5 | Detection Count=6 | Detection Count=7 | Detection Count=8 | Detection Count=9 | Detection Count=10 |
|---|---|---|---|---|---|---|---|---|
| Naïve Bayes | 79.53 | 82.62v | 84.87v | 86.65v | 88.84v | 90.73v | 91.79v | 93.45v |

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

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

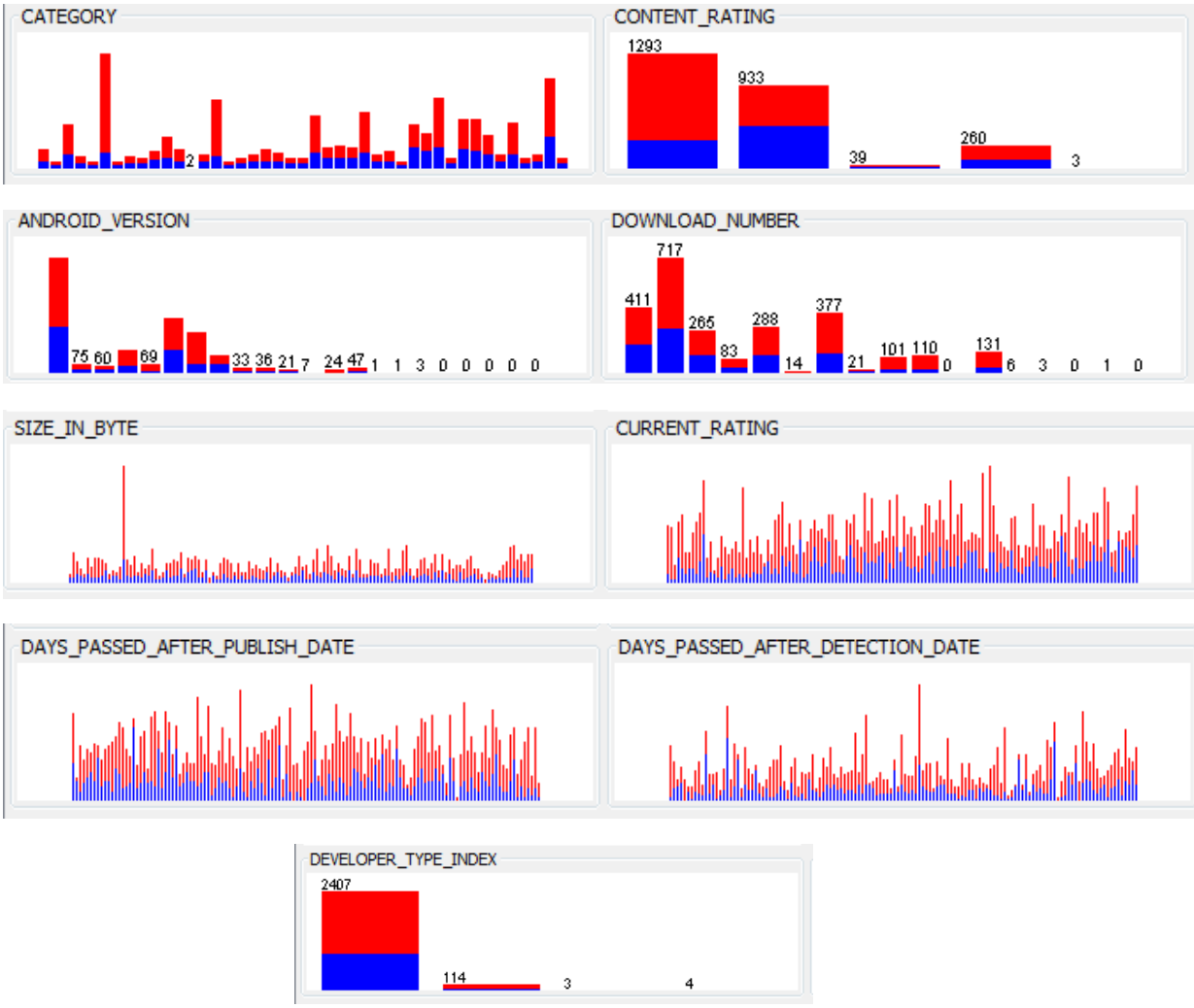
(Here the following columns are compared with the first column (dataset with detection count=3). For instance, according to the results on this table, the dataset with detection count=7 is more accurate than the dataset with detection count=3 and the difference of accuracy values between two datasets is statistically significant at 0.05 confidence value.)

Table 24-Paired t-test results for kappa statistic comparison to choose baseline datasets

| Datasets / Classifier | Detection Count=3 | Detection Count=4 | Detection Count=5 | Detection Count=6 | Detection Count=7 | Detection Count=8 | Detection Count=9 | Detection Count=10 |
|---|---|---|---|---|---|---|---|---|
| Naïve Bayes | 0.48 | 0.51 | 0.53v | 0.53v | 0.53v | 0.52v | 0.49 | 0.50 |

(Here the following columns are compared with the first column (dataset with detection count=3). For instance, according to the results on this table, the dataset with detection count=7 has higher agreement between malicious and benign applications than the dataset with detection count=3. The difference of kappa statistics between two datasets is statistically significant at 0.05 confidence value.)

## Appendix D: Histograms of Applications according to Market Related Features




*Developer name could not be visualized because it includes 1253 developers for this dataset. Red represents benign, blue represents malicious applications.

# Appendix E: Accuracy and Kappa Statistic Results of All Classification and Feature Selection Algorithms

Table 25- Accuracy and kappa statistic results for all combination of the chosen classification and feature selection algorithms, and the number of features selected by feature selection algorithms

| Classifier | Feature Selection Algorithm | # Selected Features | Results for the dataset including only permissions | | Results for the dataset including permissions and other Google Play metadata | |
|---|---|---|---|---|---|---|
| | | | Accuracy | Kappa | Accuracy | Kappa |
| NB | - | - | 75.79 | 0.45 | 80.65 | 0.57 |
| NB | CS | 10 | 75.41 | 0.45 | 83.68 | 0.64 |
| NB | CS | 30 | 75.05 | 0.45 | 80.97 | 0.58 |
| NB | CS | 50 | 75.47 | 0.45 | 80.49 | 0.57 |
| NB | IG | 10 | 75.41 | 0.45 | 83.94 | 0.65 |
| NB | IG | 30 | 75.05 | 0.45 | 81.07 | 0.59 |
| NB | IG | 50 | 75.37 | 0.45 | 80.51 | 0.57 |
| NB | RfF | 10 | 72.24 | 0.40 | 82.65 | 0.62 |
| NB | RfF | 30 | 75.09 | 0.45 | 82.39 | 0.62 |
| NB | RfF | 50 | 75.17 | 0.45 | 80.51 | 0.57 |
| J48 | - | - | 83.18 | 0.62 | 79.37 | 0.56 |
| J48 | CS | 10 | 77.55 | 0.48 | 69.69 | 0.42 |
| J48 | CS | 30 | 81.29 | 0.57 | 78.85 | 0.58 |
| J48 | CS | 50 | 82.14 | 0.59 | 73.77 | 0.47 |
| J48 | IG | 10 | 77.55 | 0.48 | 69.69 | 0.42 |
| J48 | IG | 30 | 81.44 | 0.57 | 78.85 | 0.58 |
| J48 | IG | 50 | 82.04 | 0.59 | 73.77 | 0.47 |
| J48 | RfF | 10 | 78.16 | 0.50 | 69.69 | 0.42 |
| J48 | RfF | 30 | 82.67 | 0.61 | 78.91 | 0.58 |
| J48 | RfF | 50 | 82.85 | 0.61 | 73.81 | 0.47 |
| RF | - | - | 84.81 | 0.66 | 83.66 | 0.62 |
| RF | CS | 10 | 77.82 | 0.49 | 81.74 | 0.57 |
| RF | CS | 30 | 82.57 | 0.60 | 82.28 | 0.59 |
| RF | CS | 50 | 83.44 | 0.63 | 82.85 | 0.60 |
| RF | IG | 10 | 77.76 | 0.49 | 82.93 | 0.60 |
| RF | IG | 30 | 82.95 | 0.61 | 82.59 | 0.59 |

| Classifier | Feature Selection Algorithm | # Selected Features | Results for the dataset including only permissions | | Results for the dataset including permissions and other Google Play metadata | |
|---|---|---|---|---|---|---|
| | | | Accuracy | Kappa | Accuracy | Kappa |
| RF | IG | 50 | 83.32 | 0.62 | 82.75 | 0.60 |
| RF | RfF | 10 | 77.86 | 0.50 | 81.80 | 0.57 |
| RF | RfF | 30 | 84.00 | 0.64 | 81.92 | 0.58 |
| RF | RfF | 50 | 85.30 | 0.67 | 83.26 | 0.61 |
| kNN | - | - | 83.94 | 0.64 | 85.86 | 0.69 |
| kNN | CS | 10 | 77.61 | 0.48 | 86.41 | 0.70 |
| kNN | CS | 30 | 81.92 | 0.59 | 85.70 | 0.68 |
| kNN | CS | 50 | 82.89 | 0.61 | 85.32 | 0.68 |
| kNN | IG | 10 | 77.61 | 0.48 | 86.47 | 0.70 |
| kNN | IG | 30 | 82.04 | 0.59 | 85.66 | 0.68 |
| kNN | IG | 50 | 82.85 | 0.61 | 85.40 | 0.68 |
| kNN | RfF | 10 | 77.96 | 0.50 | 84.49 | 0.66 |
| kNN | RfF | 30 | 83.34 | 0.62 | 85.82 | 0.69 |
| kNN | RfF | 50 | 84.31 | 0.65 | 86.03 | 0.69 |

## Appendix F: Paired t-test Results for Comparison of Predictive Accuracies and Kappa Statistics to Evaluate the Contribution of Official Market Metadata

Table 26- Paired t-test results for accuracy comparison to evaluate the contribution of official market metadata

| Classifier | Feature Selection Algorithm | # Selected Features | Dataset with only permissions | Dataset including permissions+ market metadata |
|---|---|---|---|---|
| Naive Bayes | - | - | 75.79 | 80.65 v |
| Naive Bayes | Chi-Square | 10 | 75.41 | 83.68 v |
| Naive Bayes | Chi-Square | 30 | 75.05 | 80.97 v |
| Naive Bayes | Chi-Square | 50 | 75.47 | 80.49 v |
| Naive Bayes | Information Gain | 10 | 75.41 | 83.94 v |
| Naive Bayes | Information Gain | 30 | 75.05 | 81.07 v |
| Naive Bayes | Information Gain | 50 | 75.37 | 80.51 v |
| Naive Bayes | ReliefF | 10 | 72.24 | 82.65 v |
| Naive Bayes | ReliefF | 30 | 75.09 | 82.39 v |
| Naive Bayes | ReliefF | 50 | 75.17 | 80.51 v |
| J48 | - | - | 83.18 | 79.37 |
| J48 | Chi-Square | 10 | 77.55 | 69.69 * |
| J48 | Chi-Square | 30 | 81.29 | 78.85 |
| J48 | Chi-Square | 50 | 82.14 | 73.77 |
| J48 | Information Gain | 10 | 77.55 | 69.69 * |
| J48 | Information Gain | 30 | 81.44 | 78.85 |
| J48 | Information Gain | 50 | 82.04 | 73.77 |
| J48 | ReliefF | 10 | 78.16 | 69.69 * |
| J48 | ReliefF | 30 | 82.67 | 78.91 |
| J48 | ReliefF | 50 | 82.85 | 73.81 |
| Random forest | - | - | 84.81 | 83.66 |
| Random forest | Chi-Square | 10 | 77.82 | 81.74 |
| Random forest | Chi-Square | 30 | 82.57 | 82.28 |
| Random forest | Chi-Square | 50 | 83.44 | 82.85 |
| Random forest | Information Gain | 10 | 77.76 | 82.93 v |
| Random forest | Information Gain | 30 | 82.95 | 82.59 |
| Random forest | Information Gain | 50 | 83.32 | 82.75 |
| Random forest | ReliefF | 10 | 77.86 | 81.8 v |
| Random forest | ReliefF | 30 | 84 | 81.92 |

| Classifier | Feature Selection Algorithm | # Selected Features | Dataset with only permissions | Dataset including permissions+ market metadata |
|---|---|---|---|---|
| Random forest | ReliefF | 50 | 85.3 | 83.26 |
| kNN | - | - | 83.94 | 85.86 |
| kNN | Chi-Square | 10 | 77.61 | 86.41 v |
| kNN | Chi-Square | 30 | 81.92 | 85.7 v |
| kNN | Chi-Square | 50 | 82.89 | 85.32 v |
| kNN | Information Gain | 10 | 77.61 | 86.47 v |
| kNN | Information Gain | 30 | 82.04 | 85.66 v |
| kNN | Information Gain | 50 | 82.85 | 85.4 v |
| kNN | ReliefF | 10 | 77.96 | 84.49 v |
| kNN | ReliefF | 30 | 83.34 | 85.82 v |
| kNN | ReliefF | 50 | 84.31 | 86.03 |

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

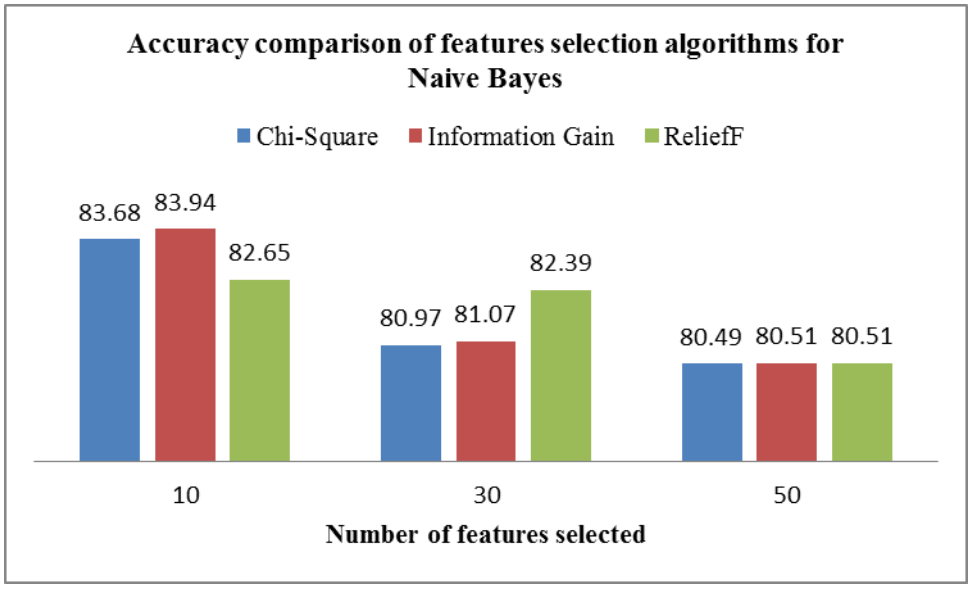
*: statistically worse than the compared value at 0.05 confidence level

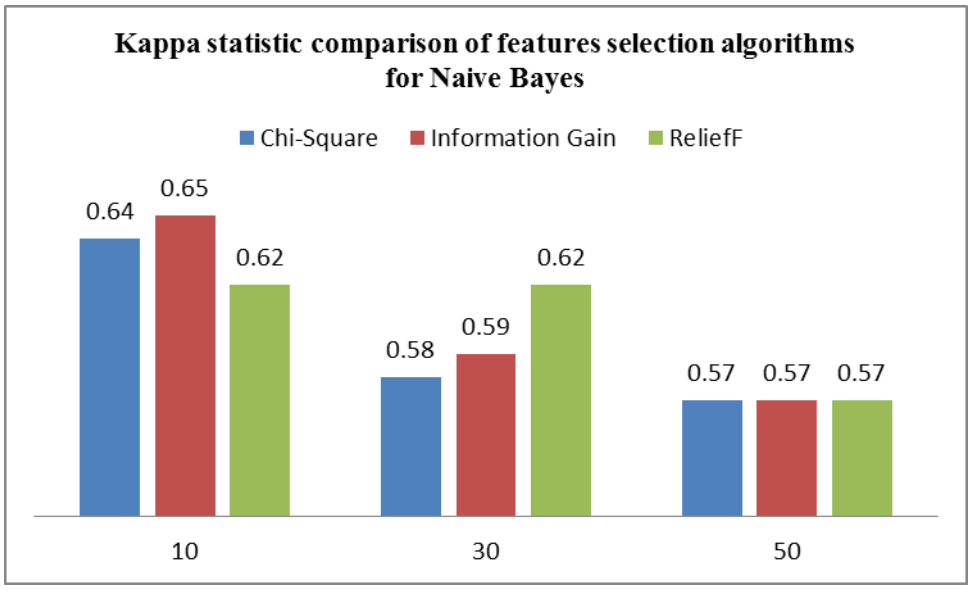
Table 27-Paired t-test results for kappa statistic comparison to evaluate the contribution of official market metadata

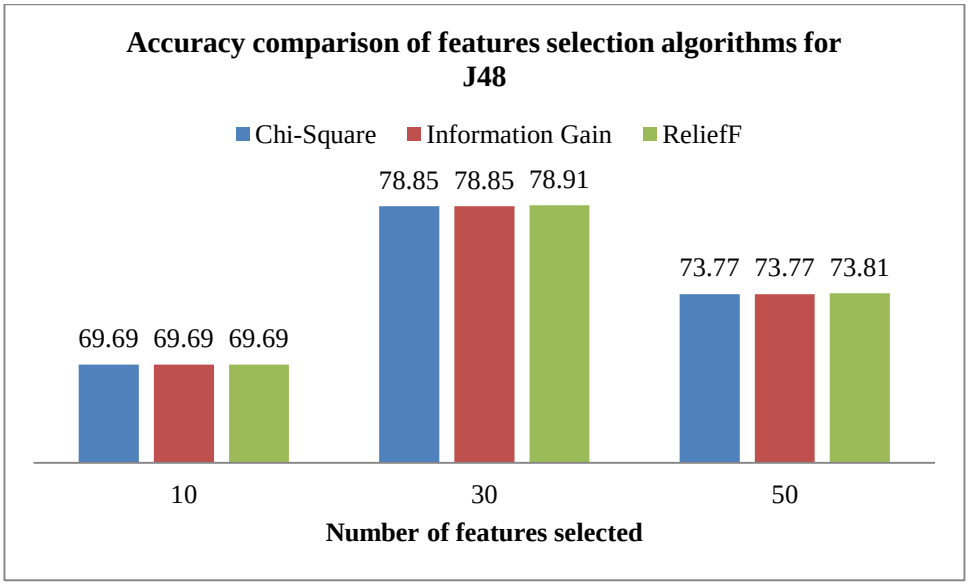
| Classifier | Feature Selection Algorithm | # Selected Features | Dataset with only permissions | Dataset including permissions+ market metadata |
|---|---|---|---|---|
| Naive Bayes | - | - | 0.45 | 0.57 v |
| Naive Bayes | Chi-Square | 10 | 0.45 | 0.64 v |
| Naive Bayes | Chi-Square | 30 | 0.45 | 0.58 v |
| Naive Bayes | Chi-Square | 50 | 0.45 | 0.57 v |
| Naive Bayes | Information Gain | 10 | 0.45 | 0.65 v |
| Naive Bayes | Information Gain | 30 | 0.45 | 0.59 v |
| Naive Bayes | Information Gain | 50 | 0.45 | 0.57 v |
| Naive Bayes | ReliefF | 10 | 0.4 | 0.62 v |
| Naive Bayes | ReliefF | 30 | 0.45 | 0.62 v |
| Naive Bayes | ReliefF | 50 | 0.45 | 0.57 v |
| J48 | - | - | 0.62 | 0.56 |
| J48 | Chi-Square | 10 | 0.48 | 0.42 |
| J48 | Chi-Square | 30 | 0.57 | 0.58 |
| J48 | Chi-Square | 50 | 0.59 | 0.47 |
| J48 | Information Gain | 10 | 0.48 | 0.42 |

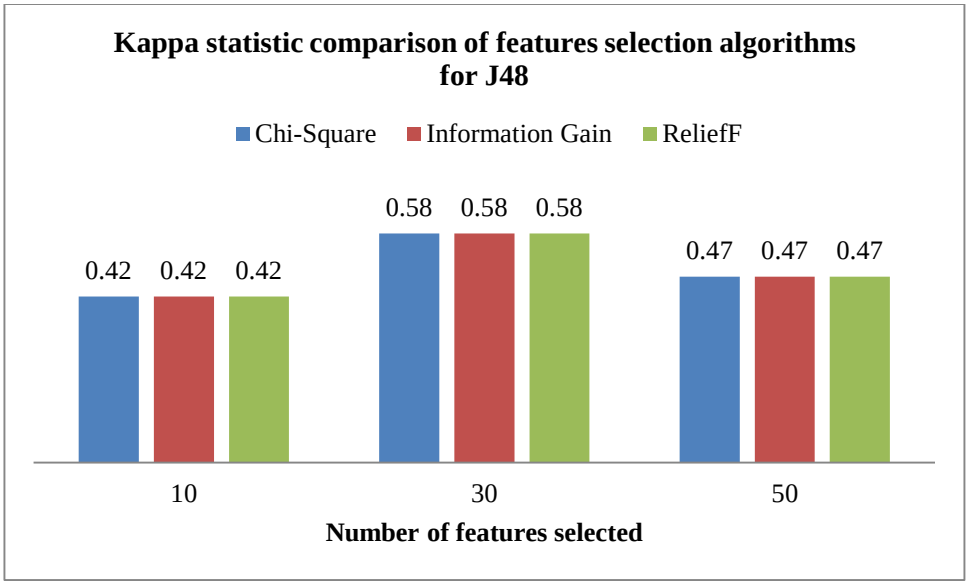
| Classifier | Feature Selection Algorithm | # Selected Features | Dataset with only permissions | Dataset including permissions+ market metadata |
|---|---|---|---|---|
| J48 | Information Gain | 30 | 0.57 | 0.58 |
| J48 | Information Gain | 50 | 0.59 | 0.47 |
| J48 | ReliefF | 10 | 0.5 | 0.42 * |
| J48 | ReliefF | 30 | 0.61 | 0.58 |
| J48 | ReliefF | 50 | 0.61 | 0.47 |
| Random forest | - | - | 0.66 | 0.62 |
| Random forest | Chi-Square | 10 | 0.49 | 0.57 |
| Random forest | Chi-Square | 30 | 0.6 | 0.59 |
| Random forest | Chi-Square | 50 | 0.63 | 0.6 |
| Random forest | Information Gain | 10 | 0.49 | 0.6 v |
| Random forest | Information Gain | 30 | 0.61 | 0.59 |
| Random forest | Information Gain | 50 | 0.62 | 0.6 |
| Random forest | ReliefF | 10 | 0.5 | 0.57 |
| Random forest | ReliefF | 30 | 0.64 | 0.58 |
| Random forest | ReliefF | 50 | 0.67 | 0.61 |
| kNN | - | - | 0.64 | 0.69 v |
| kNN | Chi-Square | 10 | 0.48 | 0.7 v |
| kNN | Chi-Square | 30 | 0.59 | 0.68 v |
| kNN | Chi-Square | 50 | 0.61 | 0.68 v |
| kNN | Information Gain | 10 | 0.48 | 0.7 v |
| kNN | Information Gain | 30 | 0.59 | 0.68 v |
| kNN | Information Gain | 50 | 0.61 | 0.68 v |
| kNN | ReliefF | 10 | 0.5 | 0.66 v |
| kNN | ReliefF | 30 | 0.62 | 0.69 v |
| kNN | ReliefF | 50 | 0.65 | 0.69 |

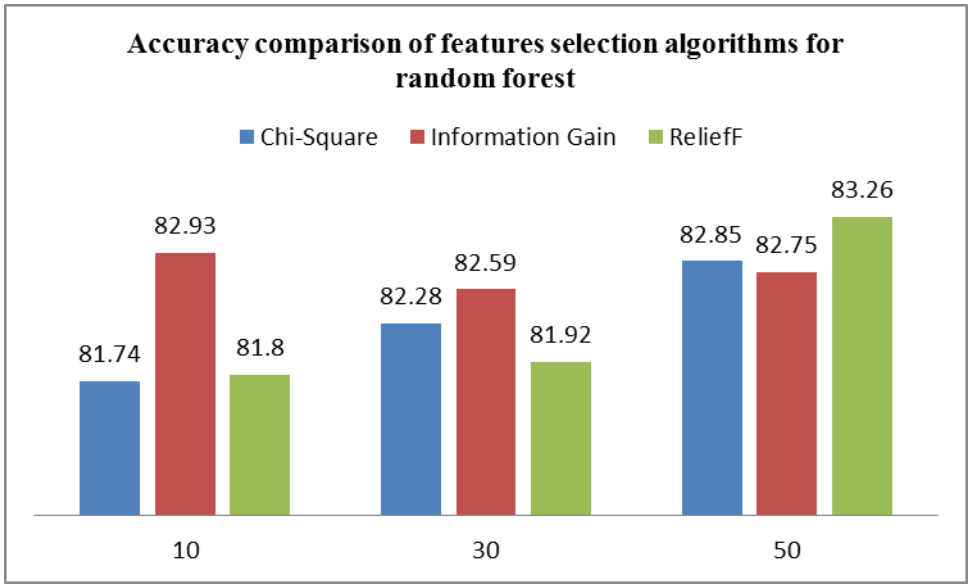
## Appendix G: Graphs for the Evaluation of Feature Selection Algorithms

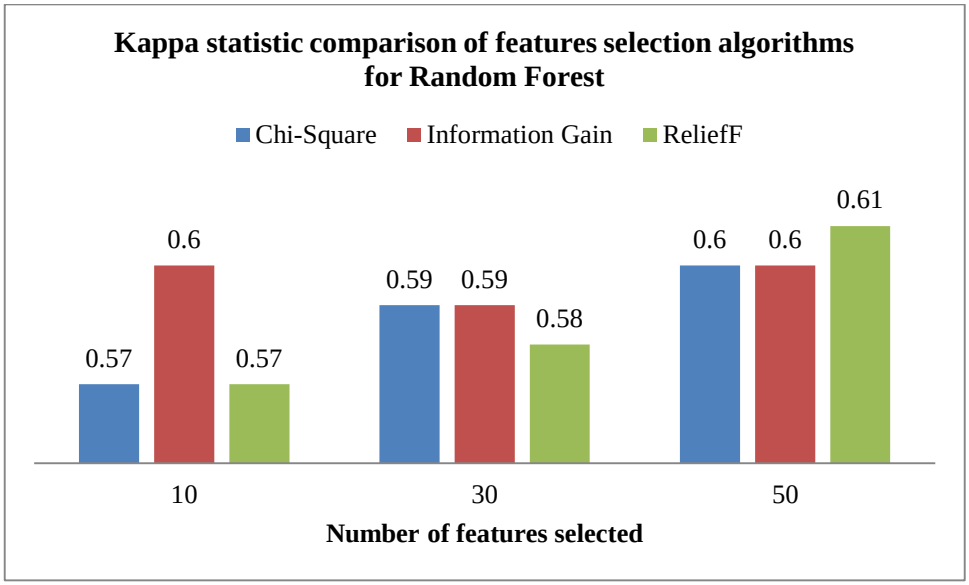


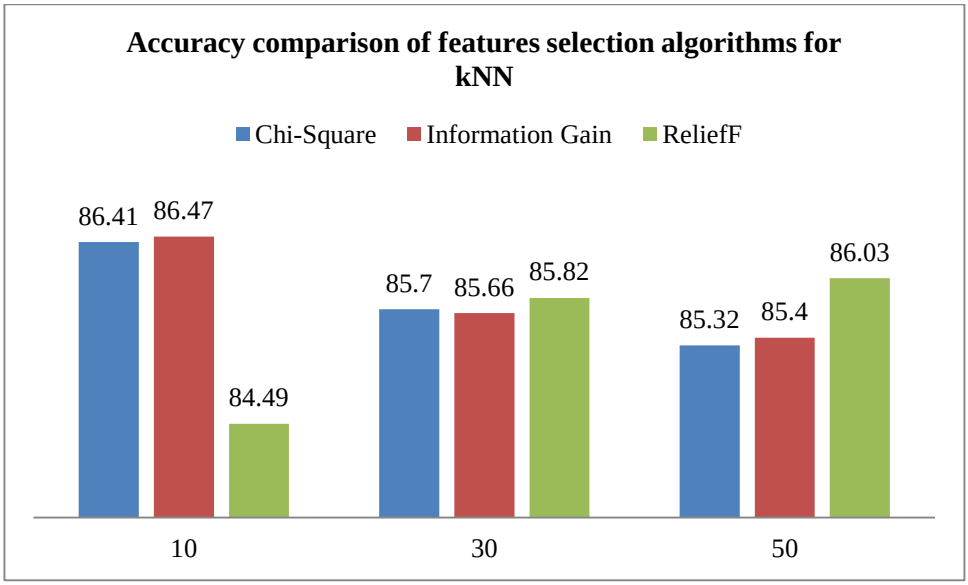



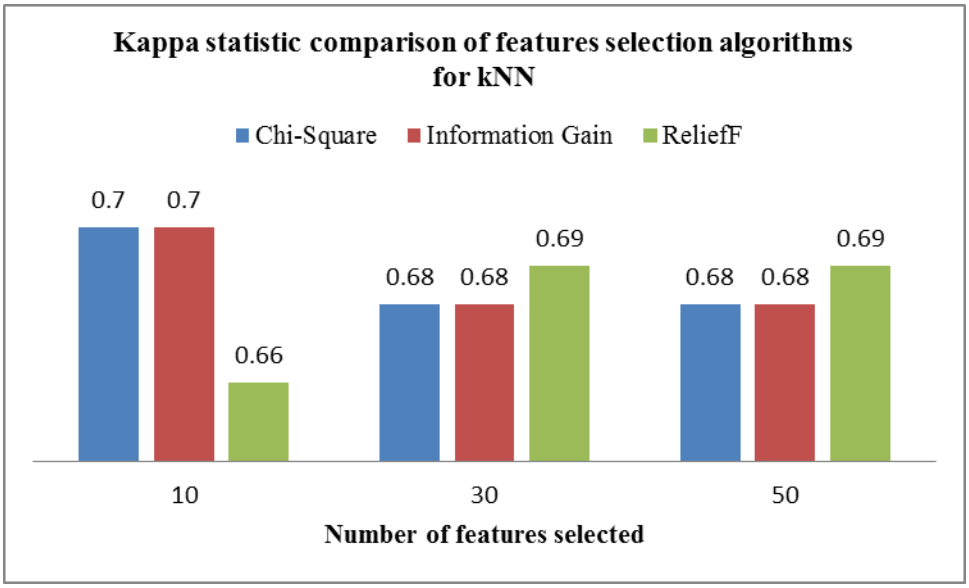

Accuracy comparison of features selection algorithms for J48
Chi-Square
Information Gain
ReliefF
69.69 69.69 69.69
78.85 78.85 78.91
73.77 73.77 73.81
10
30
50
Number of features selected


Kappa statistic comparison of features selection algorithms for J48
Chi-Square
Information Gain
ReliefF
0.42 0.42 0.42
0.58 0.58 0.58
0.47 0.47 0.47
10
30
50
Number of features selected


Accuracy comparison of features selection algorithms for random forest
Chi-Square
Information Gain
ReliefF
81.74
82.93
81.8
82.28
82.59
81.92
82.85
82.75
83.26
10
30
50


Kappa statistic comparison of features selection algorithms for Random Forest
Chi-Square
Information Gain
ReliefF
0.57
0.6
0.57
0.59
0.59
0.58
0.6
0.6
0.61
10
30
50
Number of features selected


Accuracy comparison of features selection algorithms for kNN
Chi-Square
Information Gain
ReliefF
86.41
86.47
84.49
85.7
85.66
85.82
85.32
85.4
86.03
10
30
50


Kappa statistic comparison of features selection algorithms for kNN
Chi-Square
Information Gain
ReliefF
0.7
0.7
0.66
0.68
0.68
0.69
0.68
0.68
0.69
10
30
50
Number of features selected

## TEZ FOTOKOPİSİ İZİN FORMU

**ENSTİTÜ**

Fen Bilimleri Enstitüsü ☐

Sosyal Bilimler Enstitüsü ☐

Uygulamalı Matematik Enstitüsü ☐

Enformatik Enstitüsü X

Deniz Bilimleri Enstitüsü ☐

**YAZARIN**

Soyadı : BALTACI
Adı : NURAY
Bölümü : BİLİŞİM SİSTEMLERİ

**TEZİN ADI** (İngilizce) : A COMPARISON OF CLASSIFICATION ALGORITHMS FOR MOBILE MALWARE DETECTION: MARKET METADATA AS INPUT SOURCE

**TEZİN TÜRÜ** : Yüksek Lisans X 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. X

**TEZİN KÜTÜPHANEYE TESLİM TARİHİ :** …….