Models for Risk assessment of Mobile Applications

Chigozie Charles Ikwuegbu
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Chigozie Charles Ikwuegbu
E-mail: chik19@student.bth.se

University advisor:
Dr. Anders Carlsson
Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
Abstract

**Background:** Mobile applications are software that extend the functionality our smartphones by connecting us with friends and a wide range of other services. Android, which is an operating system based on the Linux kernel, leads the market with over 2.6 million applications recorded on their official store. Application developers, due to the ever-growing innovation in smartphones, are compelled to release new ideas on limited budget and time, resulting in the deployment of malicious applications. Although there exists a security mechanism on the Google Play Store to remove these applications, studies have shown that most of the applications on the app store compromise privacy or pose security related risks. It is therefore essential to investigate the security risk of installing any of these applications on a device.

**Objectives:** To identify methods and techniques for assessing mobile application security, investigate how attributes indicate the harmfulness of applications, and evaluate the performance of K Nearest Neighbors (K-NN) and Random forest machine learning models in assessing the security risk of installing mobile applications based on information available on the application distribution platform.

**Methods:** Literature analysis was done to gather information on the different methods and techniques for assessing security in mobile applications and investigations on how different attributes on the application distribution platform indicate the harmfulness of an application. An experiment was also conducted to examine how various machine learning models perform in evaluating the security risk associated with installing applications, based on information on the application distribution platform.

**Results:** Literature analysis presents the various methods and techniques for mobile application security assessment and identifies how mobile application attributes indicate the harmfulness of mobile applications. The experimental results demonstrate the performance of the aforementioned machine learning models in evaluating the security risk of installing mobile applications.

**Conclusion:** Static, dynamic, and grey-box analysis are the methods used to evaluate mobile application security, and machine learning models including K-NN and Random forest are suitable techniques for evaluating mobile application security risk. Attributes such as the permissions, number of installations, and ratings reveal the likelihood and impact of an underlying security threat. The K-NN and Random forest models when compared to evaluate the security risk of installing mobile applications based on information on the application distribution platform showed high performance with little differences.

**Keywords:** Risk Assessment, Machine Learning, Mobile application security, Mobile application metadata.
Acknowledgments

I would like to appreciate my parents, siblings, and close relatives for their unending support throughout my studies.

I would like to thank my supervisors Anders Carlsson and Alexander Adamov for their guidance throughout my thesis.

I would like to further extend my appreciation to my professors Arkadii Snihurov, Maryna Yevdokymenko, and Amal Mersni for their continued support and encouragement throughout my thesis work.

Special thanks to Adeyemi Bernard, Ahoukpeme Amarachi, my ERASMUS family, and all my friends for their unending support and motivation.
# Contents

Abstract  

Acknowledgments  

Abbreviations  

List of figures  

List of tables  

1 Introduction  
   1.1 Problem Statement  
   1.2 Aim and Objectives  
   1.3 Research Questions  
   1.4 Decisions  
      1.4.1 Selected Application Distribution Platform  
      1.4.2 Selected Machine learning Models  
      1.4.3 Performance Metrics  
   1.5 Thesis Scope  
   1.6 Thesis Outline  

2 Background  
   2.1 Mobile Operating Systems  
   2.2 Mobile Applications  
   2.3 Machine Learning  
      2.3.1 Classification & Regression  
      2.3.2 Machine learning models  
   2.4 Security Risk Assessment  

3 Related Work  
   3.1 Similar Research Work  
   3.2 Knowledge Gap  

4 Method  
   4.1 Literature Analysis  
      4.1.1 Data Sources and Research Strategy  
      4.1.2 Criteria for selected studies  
   4.2 Experiment  
      4.2.1 Libraries  

iv
4.2.2 Experiment Design ............................................ 12
4.2.3 Used Approach .................................................. 12
4.2.4 Expert Evaluation and Data Pre-processing .................. 14
4.2.5 Implementation of the Machine learning models ............ 18
4.2.6 Evaluating the performance of the models .................. 18
4.2.7 Distribution Analysis ........................................ 19
4.2.8 Mann-Whitney U test ......................................... 19
4.3 Validity .......................................................... 20
  4.3.1 Internal Validity ............................................. 20
  4.3.2 External Validity ............................................. 20

5 Results and Analysis .............................................. 22
  5.1 Results for Literature Analysis ................................ 22
    5.1.1 Security Risk Assessment for Mobile Application .......... 22
    5.1.2 Machine learning techniques suitable for evaluating mobile application security risk .................. 23
    5.1.3 How attributes indicate the harmfulness of Mobile Applications. 24
  5.2 Experimental Results ......................................... 27
    5.2.1 Classification Results ..................................... 28
    5.2.2 Regression Results ....................................... 29
  5.3 Distribution Analysis Results .................................. 31
  5.4 Mann-Whitney U test Results .................................. 32

6 Discussion ......................................................... 33

7 Conclusions and Future Work ..................................... 34
  7.1 Conclusions .................................................... 34
  7.2 Future Work .................................................... 35

A Expert evaluation of applications security risk score and class .... 39

B Machine Learning Models .......................................... 44
  B.1 Random Forest Regression ..................................... 44
  B.2 Random Forest Classification .................................. 46
  B.3 K-NN Regression ............................................... 48
  B.4 K-NN Classification ............................................ 50

C Mann-Whitney U Test ............................................... 53
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Boolean Search Strings</td>
<td>10</td>
</tr>
<tr>
<td>4.2</td>
<td>Expert evaluation of applications</td>
<td>18</td>
</tr>
<tr>
<td>5.1</td>
<td>Mobile Application Assessment Methods</td>
<td>22</td>
</tr>
<tr>
<td>5.2</td>
<td>Android access control model [23]</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>Confusion Matrix for Machine Learning Models</td>
<td>29</td>
</tr>
<tr>
<td>5.4</td>
<td>Mean absolute error results of the models</td>
<td>30</td>
</tr>
<tr>
<td>5.5</td>
<td>Random Forest Regression Distribution Analysis</td>
<td>31</td>
</tr>
<tr>
<td>5.6</td>
<td>K-NN Regression Distribution Analysis</td>
<td>31</td>
</tr>
<tr>
<td>5.7</td>
<td>Random Forest Classification Distribution Analysis</td>
<td>32</td>
</tr>
<tr>
<td>5.8</td>
<td>K-NN Classification Distribution Analysis</td>
<td>32</td>
</tr>
</tbody>
</table>
## List of Tables

4.1 Digital Libraries. ...................................... 11  
4.2 Used Libraries. ....................................... 11  
4.3 Extracted Categories. ............................... 12  
4.4 List of Selected Applications. ...................... 13  
4.5 Details of extracted features. ...................... 14  
4.6 List of Evaluated permissions ...................... 15  
4.7 Rating Assessment. ................................... 15  
4.8 App developers Assessment. ....................... 16  
4.9 App protection level assessment. ................... 17  
4.10 App security risk level assessment. ............... 17  
4.11 Security Risk Class assessment. ................... 17  
5.1 Security assessment methods summary table ........... 23  
5.2 Machine learning techniques summary table ............. 24  
5.3 Application attribute summary table .................. 26  
5.4 Literature Analysis summary table ................... 27  
5.5 Classification Rates of the Random Forest and K-NN Models. 28  
5.6 Classification Accuracy Results of the Random Forest and K-NN Models. 28  
5.7 Random Forest and K-NN Regression Results ............ 30  
5.8 Summary of the Mann-Whitney U-test results. .......... 32  
A.1 Results of the Expert Evaluation of applications. ......... 40  
A.2 Results of the Expert Evaluation of applications(2). ......... 41  
A.3 Results of the Expert Evaluation of applications(3). ......... 42  
A.4 Results of the Expert Evaluation of applications(4). ......... 43  
C.1 Results of Regression and Classification Mann-Whitney U Test .... 53
Chapter 1

Introduction

Mobile devices are portable electronic devices or computers. These devices in recent times have high processing power, novel forms of interaction such as touch screen, new protocols of connectivity, and software that explore all these features [27].

Mobile applications (apps) are software that are downloaded unto mobile devices, tablets, and other smart devices. They extend the functionality of our smartphones to connect us with friends, give access to the internet, play games, monitor our health, and a wide range of other services [29]. These applications are issued to users through various distribution platforms defined by Pandita et al as “a central location for users to discover, purchase, download, and install software with only a few clicks within on-device market interfaces” [25]. These distribution platforms include Amazon Appstore, App Store (iOS), BlackBerry World, Google Play, and Windows store [35].

Mobile operating systems are specially made for mobile devices and make it possible for mobile applications to function. These operating systems include Symbian OS, Android OS, iOS (iPhone), BlackBerry OS, and Windows Phone [20]. Android OS, which is based on the Linux Kernel and supports third-party application development, has lead the market since 2011 with over 1.2 billion active devices as of 2015 [15]. The number of applications published on the Google Play Store expresses the demand and competition in the market with a study noting that there were more than 1.6 million apps on the Google Play Store by July 2015 and another observing the numbers to be over 2.6 million applications as of December 2016 [16, 5].

1.1 Problem Statement

Studies suggest that many of the applications on the Google Play Store suffer from some form of privacy or security risk related issue due to the time and budget limit given to developers to release new ideas on the distribution platform. This can result in the deployment of malicious applications. Although there exists a security mechanism on the Google Play Store to identify malicious applications, it is not able to inform users of the security risk associated with installing any of these applications on their device [5].

Numerous studies including that presented by Asma Hamed and Hella Kaffel Ben Ayed [29] and Haoyu et. al [4] demonstrate various techniques in identifying malicious applications with machine learning emerging as one of the popular techniques. However, the performance of various models in evaluating mobile application security risk based on app metadata is not clear. This study identifies the methods
and techniques for evaluating mobile application security, examines how mobile application attributes indicate the harmfulness of an application, and evaluates the performance of alternative machine learning models in assessing the security risk of installing mobile applications based on information available on the application distribution platform.

1.2 Aim and Objectives

The aim of the research is to evaluate the performance of alternative machine learning models in assessing the security risk of installing mobile applications on a user’s device based on information available on the distribution platform i.e. requested permissions, developers’ reputation, users ratings, etc. The results of the models will be compared with the given experts’ evaluation of the applications represented by the security risk score and class. Thus, it will be possible to evaluate the effectiveness of the proposed models.

The aim of the study is divided into the following objectives:

**O1** Identify methods and techniques for assessing security risk on mobile applications and investigate how the attributes indicate the harmfulness of applications.

**O2** Evaluate the performance of Random forest and K-Nearest Neighbors in assessing mobile application security risk.

1.3 Research Questions

The research questions to be answered in this thesis are as follows:

**RQ1.** How should security risk be assessed for mobile applications?

**RQ1.1** What machine learning techniques are suitable for evaluating mobile application security risk?

**RQ2.** How do the attributes (permissions, privacy policies, etc.) indicate the harmfulness of an application?

**RQ3.** How do different models, based on application metadata, perform when implemented to evaluate mobile application security risk?

1.4 Decisions

1.4.1 Selected Application Distribution Platform

This research evaluates the security risk of installing mobile applications from the Google Play Store. This distribution platform was selected because it is assumed that
all the applications on the Google Play Store are benign due to the existing security mechanism deployed. This platform is therefore selected in order to investigate the security risk associated with installing benign applications.

1.4.2 Selected Machine learning Models

This research compares the performance of alternative machine learning models in evaluating the security risk of installing mobile applications on a user’s device. The models to be evaluated include Random forest and K-Nearest Neighbors (K-NN). The models are selected because of their popularity and their performance as recognized by studies.

1.4.3 Performance Metrics

The performance metrics used in this study include Mean Absolute Error (MAE) and Accuracy. These metrics are used to evaluate the performance of models in regression and classification. Other metrics exist, but these were selected in order to observe the results of the research in comparison with other studies.

1.5 Thesis Scope

This thesis compares the performance of different machine learning models in evaluating the security risk of installing mobile applications based on information provided on the apps’ digital distribution platform. The performances of the various models are evaluated using the experts’ evaluation of the apps’ represented by the security risk score and class.

1.6 Thesis Outline

The remainder of the thesis is broken down into 7 chapters. Chapter 2 provides background information on various concepts adapted throughout the study. Chapter 3 discusses related works while Chapter 4 describes the methods used in the study. Chapter 5 presents the results and its analysis, while Chapter 6 covers the discussion of the results. Finally, Chapter 7 discusses the conclusions drawn, answers the research questions, and describes possible future work.
Chapter 2

Background

This chapter provides a background on various concepts adapted in the study for better understanding.

2.1 Mobile Operating Systems

Mobile operating systems are platforms made for mobile devices to run applications. These operating systems have the features of a Personal Computer (PC) operating system and determine the functions of the device. Mobile operating systems are provided by various companies with each having their own hardware and software features and they are in constant competition. These operating systems include Symbian OS, Android OS, iOS (iPhone), BlackBerry OS, and Windows Phone to name a few [2, 20].

- **Symbian OS**: The Symbian OS is currently developed by Nokia but was originally developed by Nokia, Ericsson, and Motorola. The Operating System is equipped with communication and Personal Information Management (PIM) functionalities [20].

- **Android OS**: The Android OS is based on the Linux kernel developed by Google. It supports third-party development which means it allows its users to load software from other developers. The technology is based on Java programming language and runs on devices such as Samsung, HTC, Google Pixel, and OnePlus [20, 2, 15].

- **BlackBerry OS**: is developed by Research in Motion (RIM) for the smartphone and tablets they provide. The technology is now based on Java programming language for their smart-phones and C++ on other web-based programming languages for their PlayBook tablets. The operating system provides BlackBerry Enterprise Server (BES), and BlackBerry Internet Services (BIS). These supply their devices with internet services, push-based calendar, task, contact, email, and note exchange [2].

- **iOS**: is an Operating System developed by Apple Inc for Apple devices. The technology is based on C programming language and does not allow third-party development. The Operating System supports touch-based motions such as swipe, tap, tap and hold, and squeeze to control on-screen interface components and to perform interface activities [2].
• **Windows Phone:** is an Operating System developed by Microsoft. The technology is based on C++ programming language. It is used by various manufactures such as HTC and Nokia. The newer versions of the operating system support user interface management, and a Cloud Integration module for web search via Bing, location services, and push notifications [2].

### 2.2 Mobile Applications

Mobile applications are software that are installed on mobile devices, tablets, and other smart devices and are used to perform various tasks [29]. Studies have adopted various methods in the development cycle of mobile applications, but these methods generally categorize the development process into 4 phases: identification phase, design phase, development/testing phase, and deployment phase [36].

- **Identification phase:** In this phase of the application development, information, such as the features and time needed in the development or improvement of an existing application, is discussed and documented [36].

- **Design phase:** In this phase of the development, the target distribution platform is taken into account and an initial design of the application is compiled with all the functionalities of the application broken down into various components [36].

- **Development/Testing:** In the development phase, the different components are coded and analyzed. In the testing phase, the application’s usability, functionality, and consistency is tested to ensure software confidence. This is a continuous process as the software is tested for various vulnerabilities that could be exploited and patches are provided to fix them [12, 24, 14].

- **Deployment phase:** In this phase, the application is deployed to its respective distribution platforms for users to install on their devices. An application distribution platform is a place users buy and install applications. There exist various distribution platforms and they include Amazon Appstore, App Store(iOS), BlackBerry World, Google Play, and Windows store [35, 36].

### 2.3 Machine Learning

Machine learning is a field of study which enables computers to learn from data [3]. Studies of recent use various machine learning algorithms as techniques to identify vulnerabilities in applications by extracting beneficial features. This recent trend is due to the fact that the manual extraction of patterns to identify vulnerabilities in software has proven to be tedious [37, 10].

- **Supervised Learning:** In supervised learning, the information fed to the algorithms contains the labeled output which it learns with to make predictions. Supervised machine learning models perform classification and regression tasks and some models are able to perform both tasks [3].
• **Unsupervised Learning**: In unsupervised learning, the information fed to the algorithms is not labeled and the models are left to decipher the underlying structure of the data.

### 2.3.1 Classification & Regression

• **Classification**: Classification is a supervised learning concept a model performs to group a subject into different categories. A classification problem could either be binary or multi-class. Binary classification groups data into two categories (on/off, 0/1, true/false) while multi-classification groups into multiple categories. An imbalance in classification occurs if one category has more values than the other [3].

• **Regression**: Regression unlike classification, is a concept that is used in machine learning where the models predict numerical values such as weights. [3].

### 2.3.2 Machine learning models

Machine learning has been used as a technique to identify applications that pose a security risk to users’ information and models such as random forest and k-nearest neighbors have shown high accuracy in identifying risky applications.

• **Random Forest**: is an ensemble learning method that is used for classification and regression. This method makes predictions by creating and merging multiple decision trees together and return the mode of results (for classification) and the mean (for regression) [3, 8].

• **K-Nearest Neighbors (K-NN)**: is a non-parametric learning method used for both classification and regression and does this by observing variables. The model then makes predictions based on similarity measures which account for the distance between variables [3, 8].

### 2.4 Security Risk Assessment

Risk assessment is the identification and analysis of potential events that could implicate the confidentiality, integrity, and availability of information resources on an asset and the tolerance of such an event [32]. These events in mobile applications could be as a result of vulnerabilities in the mobile application. A software vulnerability is a flaw in a system that can be exploited by an attacker [12]. Certain standards have been adopted by organizations to evaluate security risk with results produced in either a qualitative or quantitative model. These standards include OWASP, EBIOS, MEHARI, SP800-30(NIST), CRAMM, ISO27005 [24, 14].
Chapter 3

Related Work

3.1 Similar Research Work

Due to the increasing trend of malicious activities in mobile applications, studies have grown exponentially showing different approaches to assess mobile application security. This section presents the related works in the area of this thesis.

In [29] by presenting an experimental study on a set of 64 applications, Asma Hamed and Hella Kaffel Ben Ayed in their article titled “Privacy Risk Assessment and Users’ Awareness for Mobile Apps Permissions” present a scoring model that assesses the risk to users’ privacy while granting a set permissions required by an application. They observed that the number of permissions requested by an application does not influence the risk score while determining the maliciousness of an application.

Wei Wang et. al in their article titled “Detecting Android malicious apps and categorizing benign apps with an ensemble of classifiers” use 5 machine learning classifiers to categorize benign and malicious apps with the aim of easing the management of android app markets. They extract 2,374,340 features from each app to create a large dataset and use those features to detect if the app is malicious or not. In their experiment, they are able to identify malicious apps with an accuracy of 99.39%. They are also able to categorize apps with an accuracy of 82.93% [8].

Haoyu et.al in their article titled “RmvDroid: Towards A Reliable AndroidMalware Dataset with App Metadata” present an android malware dataset by taking snapshots of the Google Play Store for 4 years and using VirusTotal to tag apps with sensitive behaviors. In their approach, they monitor these apps on the Google Play Store to see if Google removes them or not. They were able to create a dataset that consists of 9,133 samples that belong to 56 malware families [4].

Jingzheng et. al in their article “PACS: Permission Abuse Checking System for Android Applications based on Review Mining” present a permission abuse system that classifies apps into different categories by considering app descriptions and users’ reviews as conditions for detection and using machine learning and data mining techniques, it mines the patterns of applications in the same category and creates a set of permissions. In their approach, they find that 726 out of 935 apps are abusing permissions [5].

In the article “Quantitative Security Risk Assessment of Android Permissions and Applications” Wang Y et. al proposed a framework for quantitative analysis of security on both android applications and permissions which was established on request patterns from both benign and malicious applications with the aim of boost-
Chapter 3. Related Work

ing the efficiency of android permission systems. In their study, they compiled two datasets which comprised of 27,274 benign apps from the Google Play Store and 1,260 android malware samples to evaluate the effectiveness of their framework [7].

Jose et. al in their article titled “Android Malware Characterization Using Metadata and Machine Learning Techniques” detect the patterns in malicious applications by analyzing indirect features and metadata of applications. In their experiment, they observe that permissions requested by an application only offer moderate performance results and that other features such as the application developer, certificate issuer, etc. are more relevant in detecting malware [6].

Pandita et. al in [25] present a framework using Natural Language Processing that identifies the need for permissions requested by an application based on the information given in the application description. Their framework achieves an average precision of 82.8% with an average recall of 81.5% for 3 permissions.

Aminordin et al. [10] in their study observed that apps in their category achieve greater accuracy compared to non-category. Their observation was based on the creation of a framework to detect malware on android applications using machine learning classifiers. In their approach, they performed a static analysis of android applications using their permissions, API calls, and the app category.

Yinglan Feng et al. in their article titled “AC-Net: Assessing the Consistency of Description and Permission in Android Apps” propose a framework which identifies the accuracy between permissions and their description by using Neural Networks, they built a dataset comprising 1,415 popular android apps and extracted the description and permissions from 11 permission groups for evaluation. Their framework records a receiver operating characteristic and a precision-recall curve value of 97.4% and 66.9% respectively [17].

Zhou et al. in their study titled “Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets” present a way of detecting malicious apps from both the official and unofficial Android markets. In their study, they crawled 5 android markets and gathered a total of 204,040 apps with the aim of detecting known and unknown malware. They present 2 methods for analysis which are permission-based and heuristics-base and their methods detected 211 malicious apps and two zero-day attacks from both official and unofficial markets [18].

Enck et al. present a framework in their research titled “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones” that monitors the behavior of 30 popular third-party android apps to identify applications that transferring data to another destination. In their study, they observed that two-thirds of the applications in their study, exhibit mishandling of sensitive data by sending precise location, phones’ unique ID, phone number, etc. due to the permissions granted during installation [13].

Jiayu Wang and Qigeng Chen Present a paper titled “ASPG: Generating Android Semantic Permissions” which aims at identifying the relationship between application description and the permission they request for. Their model is able to generate a list of permission an application needs based on the description of the application [31].
3.2 Knowledge Gap

Studies have proposed reliable approaches to detect malicious applications but to the best of the author’s knowledge, none of the identified studies evaluate the security risk of installing mobile applications considering the permissions and an extensive list of other attributes on the app distribution platform. Therefore, it is believed that the contributions of this study will be substantial to the community.
Chapter 4

Method

This chapter outlines the methods used to answer research questions tackled during the course of the thesis. To answer RQ1, RQ1.1, and RQ2, a Literature analysis was carried out and an experiment was conducted to answer RQ3.

4.1 Literature Analysis

Carrying out a literature review, the answers to RQ1, RQ1.1, and RQ2 were attained. The intention behind selecting this method was to recognize the various methods and techniques for evaluating mobile applications security, identify the machine learning technique suitable for evaluating the security risk of installing mobile applications, and investigate the application attributes that indicate the harmfulness of mobile applications with the aim of understanding how they reveal a security threat.

4.1.1 Data Sources and Research Strategy

Results that aligned to the research questions were obtained using the papers returned by the search strings in figure 4.1:

```
Figure 4.1: Boolean Search Strings
```

These strings were applied when researching for papers on digital libraries; The libraries used to extract relevant papers in the study are highlighted in table 4.1.
4.2. Experiment

Table 4.1: Digital Libraries.

<table>
<thead>
<tr>
<th>Digital Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Digital Library</td>
</tr>
<tr>
<td>BTH Summon</td>
</tr>
<tr>
<td>IEEE</td>
</tr>
<tr>
<td>Science Direct</td>
</tr>
<tr>
<td>Semantic Scholar</td>
</tr>
<tr>
<td>Springer</td>
</tr>
</tbody>
</table>

4.1.2 Criteria for selected studies

This section highlights the process used to handle the selected papers for the study. The selection was done in distinct stages. It began with reviewing the abstract and title of archived papers, followed by reading through the full texts of all the individual papers that were related to the study. Backward snowballing (a search method for extracting studies on a topic) was also used to further source out for new studies to include from the reference list of a paper.

Studies were excluded based on the following criteria:

- The paper is not published.
- The paper is not written in the English Language
- It does not present methods to assess security risk in mobile applications

4.2 Experiment

The results of RQ3 was obtained after an experiment was conducted. The motivation for choosing this method was to review how the selected attributes indicate the security risk of installing mobile applications and assess the performance of the selected techniques in evaluating the security risk of installing mobile applications. This method was also chosen due to its efficiency in providing performance measurement values as results which can be compared.

4.2.1 Libraries

The libraries used during execution are illustrated in table 4.2

Table 4.2: Used Libraries.

<table>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scikit-Learn</td>
<td>0.22.1</td>
</tr>
<tr>
<td>Pandas</td>
<td>1.0.1</td>
</tr>
<tr>
<td>Numpy</td>
<td>1.18.1</td>
</tr>
<tr>
<td>Seaborn</td>
<td>0.10.1</td>
</tr>
</tbody>
</table>
4.2.2 Experiment Design

The experiment is designed to compare the performance of the selected machine learning techniques (K-Nearest Neighbors and Random Forest models) in assessing the security risk of installing mobile applications. The models take the applications’ metadata and the security risk score and class of the applications obtained as a result of the expert evaluation and compile them into a dataset. The models are evaluated using k-fold cross-validation and analyzing the accuracy (in classification) and mean absolute error (in regression).

The independent variables of the experiment are the machine learning models while the dependent variables are the performance of the models in evaluating mobile application security risk score and class.

4.2.3 Used Approach

The experiment was carried out by creating a web scraping script\(^1\) that gathered the metadata of 60 mobile applications from the Google Play Store on 12/03/2020 and saved them in a csv file. The applications were selected from a wide range of categories shown in table 4.3. Table 4.4 shows the selected applications, Table 4.5 illustrates the details of extracted features obtained.

Table 4.3: Extracted Categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dating</td>
<td>6</td>
</tr>
<tr>
<td>Social</td>
<td>16</td>
</tr>
<tr>
<td>Entertainment</td>
<td>3</td>
</tr>
<tr>
<td>Music &amp; Audio</td>
<td>8</td>
</tr>
<tr>
<td>Shopping</td>
<td>8</td>
</tr>
<tr>
<td>Action</td>
<td>2</td>
</tr>
<tr>
<td>Arcade</td>
<td>1</td>
</tr>
<tr>
<td>Card</td>
<td>1</td>
</tr>
<tr>
<td>Racing</td>
<td>1</td>
</tr>
<tr>
<td>Roleplaying</td>
<td>1</td>
</tr>
<tr>
<td>Casual</td>
<td>1</td>
</tr>
<tr>
<td>Simulation</td>
<td>4</td>
</tr>
<tr>
<td>Word</td>
<td>1</td>
</tr>
<tr>
<td>Sports</td>
<td>1</td>
</tr>
<tr>
<td>Communication</td>
<td>4</td>
</tr>
<tr>
<td>Tools</td>
<td>2</td>
</tr>
</tbody>
</table>

\(^1\)https://github.com/Goz1/WebCrawler.git
### Table 4.4: List of Selected Applications

<table>
<thead>
<tr>
<th></th>
<th>List of Selected Applications in study</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Perfect Piano</td>
</tr>
<tr>
<td>2</td>
<td>Paint By Number - Free Coloring Book &amp; Puzzle ...</td>
</tr>
<tr>
<td>3</td>
<td>Akinator</td>
</tr>
<tr>
<td>4</td>
<td>SHEIN-Fashion Shopping Online</td>
</tr>
<tr>
<td>5</td>
<td>Love Messages for Girlfriend â†¥ Flirty Love L...</td>
</tr>
<tr>
<td>6</td>
<td>Amino: Communities and Chats</td>
</tr>
<tr>
<td>7</td>
<td>Curvy Singles Dating - Meet online, Chat &amp; Date</td>
</tr>
<tr>
<td>8</td>
<td>Sua MÃ³tica</td>
</tr>
<tr>
<td>9</td>
<td>iQIYI â€“ Movies, Dramas &amp; Shows</td>
</tr>
<tr>
<td>10</td>
<td>Amazon Music: Stream Trending Songs &amp; New Beats</td>
</tr>
<tr>
<td>11</td>
<td>AIMP</td>
</tr>
<tr>
<td>12</td>
<td>Bukalapak - Jual Beli Online</td>
</tr>
<tr>
<td>13</td>
<td>Hepsiburada</td>
</tr>
<tr>
<td>14</td>
<td>Tiki Shopping &amp; Fast Shipping</td>
</tr>
<tr>
<td>15</td>
<td>eBay Kleinanzeigen for Germany</td>
</tr>
<tr>
<td>16</td>
<td>Trendyol - HÃ±zlÃ± ve GÃ¶venli AlÃ±ÃŸveriÃŸin ...</td>
</tr>
<tr>
<td>17</td>
<td>ROMEO - Gay Dating &amp; Chat</td>
</tr>
<tr>
<td>18</td>
<td>PinkCupid - Lesbian Dating App</td>
</tr>
<tr>
<td>19</td>
<td>EliteSingles: Dating App for singles over 30</td>
</tr>
<tr>
<td>20</td>
<td>Sverige Chat &amp; Dating</td>
</tr>
<tr>
<td>21</td>
<td>Bale</td>
</tr>
<tr>
<td>22</td>
<td>WIPPY - When you need a friend nextdoor</td>
</tr>
<tr>
<td>23</td>
<td>VK â€” live chatting &amp; free calls</td>
</tr>
<tr>
<td>24</td>
<td>Telegram X</td>
</tr>
<tr>
<td>25</td>
<td>Facebook Lite</td>
</tr>
<tr>
<td>26</td>
<td>All social media and social networks in one app</td>
</tr>
<tr>
<td>27</td>
<td>Snapchat</td>
</tr>
<tr>
<td>28</td>
<td>TikTok</td>
</tr>
<tr>
<td>29</td>
<td>Mad For Dance - Taptap Dance</td>
</tr>
<tr>
<td>30</td>
<td>Instagram</td>
</tr>
<tr>
<td>31</td>
<td>House Flip</td>
</tr>
<tr>
<td>32</td>
<td>Scrabble® GO - New Word Game</td>
</tr>
<tr>
<td>33</td>
<td>Moto Rider GO: Highway Traffic</td>
</tr>
<tr>
<td>34</td>
<td>Bullet League - 2D Battle Royale</td>
</tr>
<tr>
<td>35</td>
<td>Idle War: Legendary Heroes</td>
</tr>
<tr>
<td>36</td>
<td>Real Boxing â€“ Fighting Game</td>
</tr>
<tr>
<td>37</td>
<td>Dragon City - Collect, Evolve &amp; Build your Island</td>
</tr>
<tr>
<td>38</td>
<td>Dice Dreams</td>
</tr>
<tr>
<td>39</td>
<td>Property Brothers Home Design</td>
</tr>
<tr>
<td>40</td>
<td>Snake.io - Fun Addicting Arcade Battle .io Games</td>
</tr>
<tr>
<td>41</td>
<td>Cooking Fever</td>
</tr>
<tr>
<td>42</td>
<td>Solitaire Showtime: Tri Peaks Solitaire Free &amp;...</td>
</tr>
</tbody>
</table>
Table 4.4 – continued from previous page

<table>
<thead>
<tr>
<th></th>
<th>List of Selected Applications in study</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>Free QR code Scanner app</td>
</tr>
<tr>
<td>44</td>
<td>Radyo Kulesi - Turkish Radios</td>
</tr>
<tr>
<td>45</td>
<td>Smule - The Social Singing App</td>
</tr>
<tr>
<td>46</td>
<td>True ID Caller Name: Caller ID, Call Block</td>
</tr>
<tr>
<td>47</td>
<td>AliExpress - Smarter Shopping, Better Living</td>
</tr>
<tr>
<td>48</td>
<td>Kate Mobile for VK</td>
</tr>
<tr>
<td>49</td>
<td>OK</td>
</tr>
<tr>
<td>50</td>
<td>Samsung music</td>
</tr>
<tr>
<td>51</td>
<td>My Mixtapez Music</td>
</tr>
<tr>
<td>52</td>
<td>11st</td>
</tr>
<tr>
<td>53</td>
<td>Imo</td>
</tr>
<tr>
<td>54</td>
<td>Video chat - Oz Cam</td>
</tr>
<tr>
<td>55</td>
<td>Super Backup &amp; Restore</td>
</tr>
<tr>
<td>56</td>
<td>Nyango: Cheap VoIP International Mobile, Call I...</td>
</tr>
<tr>
<td>57</td>
<td>MobileVOIP Cheap international Calls</td>
</tr>
<tr>
<td>58</td>
<td>DateU Pro - Meet, Love &amp; Date</td>
</tr>
<tr>
<td>59</td>
<td>WAVE - Video Chat Playground</td>
</tr>
<tr>
<td>60</td>
<td>SMOOTHY - Group Video Chat</td>
</tr>
</tbody>
</table>

Table 4.5: Details of extracted features.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of App</td>
<td>Name of application</td>
<td>Alphabetical</td>
</tr>
<tr>
<td>Rating</td>
<td>Ratings of applications</td>
<td>Numeric</td>
</tr>
<tr>
<td>No of Installs</td>
<td>Indicates the number of installs</td>
<td>Numeric</td>
</tr>
<tr>
<td>Offered By</td>
<td>Showing the developers of the app</td>
<td>Alphabetical</td>
</tr>
<tr>
<td>Website</td>
<td>Gives more detail about the developer(s)</td>
<td>Alphabetical</td>
</tr>
<tr>
<td>Privacy Policy</td>
<td>Describes the policy of the application</td>
<td>Alphabetical</td>
</tr>
<tr>
<td>email</td>
<td>Contact of the developer(s)</td>
<td>Alphabetical</td>
</tr>
<tr>
<td>Permissions</td>
<td>Requested permissions of applications</td>
<td>Alphabetical</td>
</tr>
</tbody>
</table>

4.2.4 Expert Evaluation and Data Pre-processing

The expert evaluation presents the assessment of applications which determined the security risk score and class of an application using risk assessment formulas. It was implemented in an excel file taking into account the commonly requested permissions as considered by recent studies [7, 17, 6] and shown in table 4.6, ratings, developers’ details, and the number of installs of each application and assigning them two values: a risk factor according to assessment methods, and a weight factor between (0.0 and 1.0) which described the value of the attribute when evaluating security risk for mobile applications.
### Table 4.6: List of Evaluated permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Protection Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS_COARSE_LOCATION</td>
<td>Dangerous</td>
</tr>
<tr>
<td>ACCESS_FINE_LOCATION</td>
<td>Dangerous</td>
</tr>
<tr>
<td>ACCESS_LOCATION_EXTRA_COMMANDS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>ACCESS_MEDIA_LOCATION</td>
<td>Dangerous</td>
</tr>
<tr>
<td>ACCESS_BACKGROUND_LOCATION</td>
<td>Dangerous</td>
</tr>
<tr>
<td>READ_CONTACTS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>READ_PHONE_NUMBERS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>WRITE_CONTACTS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>RECORD_AUDIO</td>
<td>Dangerous</td>
</tr>
<tr>
<td>BLUETOOTH</td>
<td>Normal</td>
</tr>
<tr>
<td>BLUETOOTH_ADMIN</td>
<td>Normal</td>
</tr>
<tr>
<td>SEND_SMS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>READ_CALL_LOG</td>
<td>Dangerous</td>
</tr>
<tr>
<td>WRITE_CALL_LOG</td>
<td>Dangerous</td>
</tr>
<tr>
<td>ANSWER_PHONE_CALLS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>CALL_PHONE</td>
<td>Dangerous</td>
</tr>
<tr>
<td>READ_CALENDAR</td>
<td>Dangerous</td>
</tr>
<tr>
<td>WRITE_CALENDAR</td>
<td>Dangerous</td>
</tr>
<tr>
<td>ACCESS_WIFI_STATE</td>
<td>Normal</td>
</tr>
<tr>
<td>CAMERA</td>
<td>Dangerous</td>
</tr>
<tr>
<td>BODY_SENSORS</td>
<td>Dangerous</td>
</tr>
<tr>
<td>READ_EXTERNAL_STORAGE</td>
<td>Dangerous</td>
</tr>
<tr>
<td>WRITE_EXTERNAL_STORAGE</td>
<td>Dangerous</td>
</tr>
</tbody>
</table>

The ratings of applications give a user the numeric estimation of what others think of the said application. During the assessment, the ratings were assigned a risk factor as shown in table 4.7 and a weight factor of (0.1).

### Table 4.7: Rating Assessment.

<table>
<thead>
<tr>
<th>Ratings value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 - 3.0</td>
<td>An application with these ratings is not trustworthy</td>
</tr>
<tr>
<td>3.1 - 5.0</td>
<td>An application with these ratings is trustworthy</td>
</tr>
</tbody>
</table>

The number of installations gives a user an idea of the number of people that trust the application well enough to install it on their device. Using the computation described in 4.2, the numbers were scaled down to be in the range between (1 and 10). Applications that fell between (5 and 10) after the scaling was assigned the risk factor of (0) while others were assigned (1) and the weight factor of (0.1).

It is understood that developers spend a fortune to secure their apps and to assess the developer(offered by), a white-list comprising of developers/companies on the Forbes list [28] was put together and applications were assigned a risk factor
(according to the table 4.8) as a result of the companies/developers resources to better secure their applications during its life cycle. The attribute was then assigned a weight factor of (0.1) for each application.

Table 4.8: App developers Assessment.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Default values for applications</td>
</tr>
<tr>
<td>0</td>
<td>If the application is developed by a well-known developer/company (derived from Forbes list of top companies)</td>
</tr>
</tbody>
</table>

The developers’ details give more information about the developer(s) to a user. App developers that provide information such as privacy policy, email address, and website give users a means to reach them and can be deemed trustworthy and transparent. These apps were assigned the risk factor of (0) while others without such information were assigned (1) and a weight factor of (0.1) during the assessment of each developers’ detail.

Applications come with varying numbers of permissions that are used to enforce access control. Therefore to get the first value, each permission was accessed taking into account the following:

- The permissions impact on confidentiality, integrity, and availability of information (CIA).
- The permissions protection levels assigned by android systems as illustrated in table 4.9.
- The security risk level describes how many resources the permission controls as shown in table 4.10.

And was computed using the formula in 4.1 as described by [14, 33]. The results were then normalized to fit a range from (0, 100) using the computation in 4.2 and assigned a weight factor of (0.4) as the second value for each permission.

\[
P_i = \sum (I_i \times T_i \times S_i)
\]  \hspace{1cm} (4.1)

Where \( I_i \) is the impact of each permission on Confidentiality, Integrity, and Availability, \( T_i \) is the protection level and \( S_i \) is the security risk level.

\[
P'_i = a + \frac{(P_i - P_i(min))(b - a)}{P_i(max) - P_i(min)}
\]  \hspace{1cm} (4.2)

Where \( P'_i \) is the normalized value and \([a, b]\) is the range.

Once all the values were assigned, the risk score of an application was evaluated by multiplying the risk factor with the weight of factor for each application and summing up the values for all the attributes as illustrated in the computation 4.3:

\[
R_i = \sum (A_i \times W_i)
\]  \hspace{1cm} (4.3)
Table 4.9: App protection level assessment.

<table>
<thead>
<tr>
<th>Protection level</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>Dangerous</td>
<td>2</td>
</tr>
<tr>
<td>Signature</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.10: App security risk level assessment.

<table>
<thead>
<tr>
<th>Sec. Risk level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The permission allows access to a specific resource</td>
</tr>
<tr>
<td>2</td>
<td>The permission allows access to more than one resource</td>
</tr>
<tr>
<td>3</td>
<td>The permission allows access to multiple resources</td>
</tr>
</tbody>
</table>

Where $A_i$ is the risk factor for each application, and $W_i$ is the weight of the attribute and the sum of the weights equals 1.

Taking account of the risk score assigned to each application, a class of either high, medium, or low was then assigned to an application as illustrated in table 4.11. The computation in 4.4 was used to ensure the scores were split into thirds. Figure 4.2 shows the application attribute, risk score and class of 9 applications. The full list can be found in appendix A.

$$Q = q(n + 1)$$ (4.4)

Where $q$ is the quantile, the proportion below the $ith$ value and $n$ is the number of observations.

Table 4.11: Security Risk Class assessment.

<table>
<thead>
<tr>
<th>Class</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>$R_i &gt; 20.0$</td>
</tr>
<tr>
<td>Medium</td>
<td>$9.5 &lt; R_i \leq 20.0$</td>
</tr>
<tr>
<td>Low</td>
<td>$0 \leq R_i \leq 9.5$</td>
</tr>
</tbody>
</table>

Where $R_i$ is the risk score of an application.
4.2.5 Implementation of the Machine learning models

The machine learning models were implemented in python using scikit-learn library and comprised K-Nearest Neighbors and Random Forest implementations.

The models take the application attributes represented with the risk score and class as variables and by encoding nominal labels, convert non-numerical values. If the applications provide a privacy policy, email address, and a website, they were assigned the values (1) else, they were set to (0). The number of installs and ratings for applications were scaled down as described in the expert evaluation. The developer(offered by) was also labeled as explained during the assessment. If the application had permissions corresponding to those being considered, they were assigned the value (1) else they were assigned (0) after which they were split into training and testing sets to evaluate their performance in classification and regression.

4.2.6 Evaluating the performance of the models

To assess the models, grid search hyperparameter tuning (which outputs the best parameters for a model by running through a set of manually specified values) was used since the models should return their best performances for regression and classification. This was followed by 10-fold cross-validation. The k-fold cross-validation is a re-sampling method used to evaluate the effectiveness of a model on unseen data. This method divides the set into a number of folds and handles the first fold as a validation set and fits the remaining to the first fold [3].

In evaluating the efficiency of the models in regression analysis, the mean absolute error(The average sum of all absolute errors) of the models were analyzed using the computation in 4.5.

\[ MAE = \frac{\sum_{i=1}^{n} abs(y_i - \lambda(x_i))}{n} \]  

Where \( y_i \) is the prediction and \( \lambda(x_i) \) is the true value.

To evaluate the accuracy of the models in classifications, the computation in 4.6 was used taking into account the following:
4.2. Experiment

- **True Positive Rate**: describes the rate the values are correctly predicted for a class.

- **True Negative Rate**: describes the rate the incorrect values are identified for a class.

- **False Positive Rate**: describes the rate incorrect values are predicted as correct for a class.

- **False Negative Rate**: describes the rate correct values were predicted as incorrect for a class.

\[
\begin{align*}
TP_i &= \frac{TP_i}{TP_i + TN_i} \\
TN_i &= \frac{TN_i}{TN_i + FP_i} \\
FP_i &= \frac{FP_i}{FP_i + TN_i} \\
FN_i &= \frac{FN_i}{FN_i + TP_i}
\end{align*}
\]

\[
Accuracy = \frac{TP_i + TN_i}{k}
\]

Where \( k \) is the total number of classes, \( TP \) is the true positive, \( TN \) is the true negative value, \( FP \) is the false positive values, and \( FN \) which is the false negative.

4.2.7 Distribution Analysis

To evaluate the significance in the samples, a hypothesis test was carried out using the t-test which is a parametric test that identifies if there is a significant difference between samples. However, the test assumes the distribution of samples is normal and sample distribution analysis was used to examine if the assumption holds. Histograms and QQ plots of the samples were then created.

4.2.8 Mann-Whitney U test

The data samples of the experiment were compared using a Mann-Whitney U test. A Mann-Whitney U test is a nonparametric test used to compare significance in samples by ranking them and determining if the two sets are mixed randomly or gathered on opposite ends. The test is performed on the samples using the formula 4.7:

\[
\begin{align*}
U_1 &= n_1n_2 + \frac{n_1(n_1 + 1)}{2} - R_1 \\
U_2 &= n_1n_2 + \frac{n_2(n_2 + 1)}{2} - R_2
\end{align*}
\]
Where \( n_1 \) and \( n_2 \) are the sample sizes, \( R_1 \) and \( R_2 \) are the ranks of the samples.

Moving forward, the two-tailed probability estimate, the p-value is realized based on the U-values calculated. The significance level \( \alpha \) is set to 0.05. This will ensure that the statistical difference (if any is observed) did not occur by chance. If \( p > \alpha \) the null hypothesis is not rejected else if \( p < \alpha \) the null hypothesis will be rejected.

The research hypothesis is stated as follows:

\( H_0: \) There are no significant differences between the models when implemented to assess mobile application security risk based on publicly available information.

\( H_a: \) There is a significant difference between the models when implemented to assess mobile application security risk based on publicly available information.

4.3 Validity

4.3.1 Internal Validity

Literature Analysis

The search strings used in gathering articles did not produce all the studies in the research area and to battle this issue, snowballing was combined with the search to improve the discovery of studies.

Experiment

The time used in gathering the attributes of mobile applications for assessment could affect the results of the experiment. If the metadata of some applications were manually documented on a certain date and others were documented on a later date, the attribute values such as rating and number on installations might differ from what they would have been if all were documented on the same day. To eliminate this issue, a script was written which collected all the metadata on the same day and saved them.

4.3.2 External Validity

Firstly, the applications studied only cover a small fraction of applications that can be found on the Google Play Store. To reduce this threat, the applications were selected from different categories as shown in figure 4.3 to guarantee wide coverage.

Furthermore, the expert evaluation of the applications was carried out by the author which could lead to bias of the results. To reduce this bias and ensure generalization, the expert evaluation of the application attributes adopted risk assessment methods such as that provided by [1] and took account of the permissions protection level assigned by Android. The machine learning models were implemented using the mentioned libraries to improve external validity as it makes it possible to execute outside the experiment. The implementation source code was vetted across documentation to ensure accuracy. The size of the samples amongst other factors makes it challenging to determine the type of statistical test to employ. Using the sample
distribution test and analyzing the Histogram and QQ plots of the samples improves the results.
This chapter presents the results of the study. The results of the literature analysis are presented in 5.1 and the results of the experiment are presented in 5.2.

5.1 Results for Literature Analysis

The results of the literature analysis are presented in 5.1.1, 5.1.2, and 5.1.3 giving answers to RQ1, RQ1.1, and RQ2. Table 5.4 presents a list showing the articles used for the literature analysis.

5.1.1 Security Risk Assessment for Mobile Application.

Studies identify software vulnerabilities and evaluate the security risk of mobile applications using various assessment methods. These methods are generally categorized into static analysis, dynamic analysis, and grey-box analysis as shown in figure 5.1. Table 5.1 presents a summary of the studies that answered the research question.

Static Analysis

Static Analysis, also called white-box analysis, is the assessment of the source code of software for vulnerabilities and is carried out without the execution of the program [19, 38]. This method allows for careful evaluation of the structure of the program to review it for privacy data leaks, permission misuse, vulnerabilities, and code verification [11]. It does not depend on compilers and compiler environments making
it possible to find errors that might surface years later. Control-flow analysis, data-flow analysis, lexical analysis, and static taint analysis are techniques that fall under static analysis [30, 12, 37, 38, 34, 37].

 Dynamic Analysis

Dynamic Analysis, also known as black-box analysis, is another method of assessing mobile applications for security flaws. It requires the execution of the program being tested in a sandbox which replicates the exact environment of the application. Various attacks are then simulated and the software is monitored to identify where it might default. Techniques that adopt dynamic analysis include dynamic taint analysis and behavioral analysis [19, 38, 34, 37, 9].

 Grey-Box Analysis

Grey box analysis is a hybrid method used to identify software flaws and is a combination of both static and dynamic. In this method of testing, the source code of the application is partially known to allow for better development of test cases. Techniques that adopt dynamic analysis include regression analysis, pattern testing, and matrix testing [11, 38].

 Table 5.1: Security assessment methods summary table

<table>
<thead>
<tr>
<th>Security assessment Methods</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Analysis</td>
<td>[30], [12], [37], [38], [34], [37], [19], [11]</td>
</tr>
<tr>
<td>Dynamic Analysis</td>
<td>[19], [38], [34], [37], [9]</td>
</tr>
<tr>
<td>Grey-Box Analysis</td>
<td>[11], [38]</td>
</tr>
</tbody>
</table>

 5.1.2 Machine learning techniques suitable for evaluating mobile application security risk.

Machine learning has been used in various studies as a technique for analyzing software security due to the difficulty of manually specifying and updating patterns for other analysis methods. This has been attained by extracting static and dynamic features and using them to train various machine learning models. The models identified as a result of the literature analysis include random forest, k-nearest neighbors, support vector machine (SVM), logistic regression, decision trees [22, 8, 37, 15, 10, 9]. This study evaluates the performance of k-nearest neighbors and random forest due to their performance and popularity as identified by studies. Machine learning technique was therefore selected to examine its performance in evaluating security and comparing the results with other studies. Table 5.2 presents a summary of the studies that answered the research question.
Table 5.2: Machine learning techniques summary table

<table>
<thead>
<tr>
<th>Machine learning techniques</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine (SVM)</td>
<td>[22], [8], [37], [15], [10]</td>
</tr>
<tr>
<td>K Nearest Neighbors (K-NN)</td>
<td>[9], [8]</td>
</tr>
<tr>
<td>Logistics regression</td>
<td>[15]</td>
</tr>
<tr>
<td>Decision trees</td>
<td>[9], [15], [10]</td>
</tr>
<tr>
<td>Random Forest</td>
<td>[9], [8], [10]</td>
</tr>
</tbody>
</table>

5.1.3 How attributes indicate the harmfulness of Mobile Applications.

Mobile application attributes such as the permissions, the application description, review, number of installs, ratings, privacy policy, the developer, and the developers’ details uncover the likelihood of a security threat and the impact of the security threat.

Permissions

The permissions in the Android system are used to enforce access control i.e they are used to access resources such as files and access to network settings. For mobile applications to carry out their functions, they make requests for these permissions which include API calls and file system permissions from a user’s device [5, 29, 39].

Android provides a security model to govern how mobile application developers make such requests via the use of a permission model as shown in figure 5.2. This model compels the app developers to declare the permissions they need to run their apps in the “Manifest.xml file” of their application which the app users will see during install-time (before the application starts running for android version 5.1 and below) or run-time (while the application is being installed for android version 6.0 and higher) and are free to opt-out from granting them. However, most users ignore or do not understand the impact of these permissions which could lead to permission abuse, information leakage and many other malicious activities [5, 23, 39].

In investigating the permissions of an application, the amount of information that could be exposed and the affected resources in the case of a security breach can be revealed [5, 23, 39, 29, 37].

- **Normal**: These are permissions that pose very little risk to a users’ privacy or operations on other apps and are granted automatically by the system [21].

- **Dangerous**: These are those permissions that requests for data or resources that involve the users’ personal information and pose a high risk to the user. These permissions can read and potentially write users’ data [21, 17].

- **Signature**: These are permissions that are granted by the system during install time, but only does so when the application requesting them is signed by the same certificate as the application [21].
5.1. Results for Literature Analysis

Figure 5.2: Android access control model [23]

**Application Description**

The application description gives users information about the application. Application developers are presented with this field to give structured information of the application when launching it on the application distribution platform.

In reviewing the application description of an application, the functionality of the application can be uncovered revealing the information that will be needed from users shedding light on the resources that could be affected in the case of a security incident [5, 17].

**Application Review**

The application review is a description of what users think about an application. Users of an application are encouraged to review an application giving their opinion of the application and this detail is displayed on the application distribution platform.

Investigating the review of an application reveals how trustworthy an application is [5, 6].

**Number of Installations**

The number of installations of an application is a numeric description detailing the number of users of an application. This information resides on the application distribution platform and reviewing this detail describes the number of people that trust an application well enough to install it on their device. It also describes the number of people that will be infected in the case of a security incident [6].

**Ratings**

The application ratings are a numeric review of an application. Users of an application are encouraged to rate an application and though this is done once, they are free to update their opinion anytime and the average of all users’ ratings is displayed on the application distribution platform.

Investigating the ratings of an application reveals how trustworthy an application is [5, 6].
Privacy Policy

The privacy policy of an application is provided by the application developer. This document is provided by the application developers for their users giving details of the information to be gathered from a user and how the this information is to be used.

Reviewing the privacy policy of the application examines how transparent the developers with what they require from the users which signifies how trustworthy they are [6].

The Developers’ Details

The developers’ details include attributes such as the website and email address which the application developers provide on the application distribution platform for application users to get more information about them and how to get across to them. These attributes express the transparency of the application developers showing how trustworthy they are [6].

The Offered By

The application developer (offered by) presents the name of the developer(s) or the company offering an application. Reviewing the developer makes it possible to investigate how trustworthy a developer is. By reviewing the success of their other applications, investigating if their other applications have had issues with security in the past or reviewing the status of the developer(s)/company the likelihood of a security incident can be determined [6].

Studies have mainly focused on permissions in training machine learning models to identify risky applications. This study, however, investigates the permissions, number of installations, ratings, privacy policy, developers’ details, and offered by to investigate how they aid in identifying risky applications. Table 5.3 presents a summary of the studies that answered the research question.

Table 5.3: Application attribute summary table

<table>
<thead>
<tr>
<th>Mobile application attributes</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>likelihood and impact of a security threat</td>
<td>[5], [29], [39], [23], [37], [17], [6]</td>
</tr>
</tbody>
</table>
Table 5.4: Literature Analysis summary table.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Research Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Yes</td>
</tr>
<tr>
<td>19</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>Yes</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
</tr>
<tr>
<td>39</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>Yes</td>
</tr>
<tr>
<td>37</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
</tr>
<tr>
<td>22</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>-</td>
</tr>
<tr>
<td>38</td>
<td>Yes</td>
</tr>
<tr>
<td>34</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

5.2 Experimental Results

The results of the experiment are presented in 5.2.1, and 5.2.2 giving answers to RQ3. The result of the hypothesis test is presented in 5.3.
5.2.1 Classification Results

The results of the models in classification can be seen in table 5.5 and 5.6. They show the True Positive, True Negative, False Negative, False Positive, and accuracy of the models in predicting risk score and class. Figure 5.3 presents the confusion matrix plot of the models.

Table 5.5 shows the rates of the models’ predictions. Table 5.6 shows the accuracy of the models in predicting the security risk class of applications (Classification). The results show that the K-Nearest Neighbors model predicts the class of applications with 85% accuracy while the Random Forest model predicts the class with 82% accuracy.

Figure 5.3 shows the confusion matrix plot of the classification results which highlights the true positive, true negative, false positive, false negative rates of the models. 5.3a illustrates the Random forest results and the 5.3b illustrates the K-NN results with the predicted values on the x-axis and the true values on the y-axis.

Table 5.5: Classification Rates of the Random Forest and K-NN Models.

<table>
<thead>
<tr>
<th>Models</th>
<th>True Positive</th>
<th>True Negative</th>
<th>False Positive</th>
<th>False Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.85</td>
<td>0.65</td>
<td>0.95</td>
<td>0.85</td>
</tr>
<tr>
<td>K-Nearest Neighbors</td>
<td>0.8</td>
<td>0.75</td>
<td>1.0</td>
<td>0.975</td>
</tr>
</tbody>
</table>

"H" refers to the High-class risk, "M" refers to the Medium-class risk and "L" refers to the Low-class risk.

Table 5.6: Classification Accuracy Results of the Random Forest and K-NN Models.

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>0.8166666666666667</td>
</tr>
<tr>
<td>K-Nearest Neighbors</td>
<td>0.85</td>
</tr>
</tbody>
</table>
5.2. Experimental Results

(a) Random Forest Classification

(b) K-NN Classification

Figure 5.3: Confusion Matrix for Machine Learning Models

5.2.2 Regression Results

The results of the models in regression can be seen in table 5.7 and figure 5.4.

Table 5.7 shows the mean absolute error of the models in predicting the security risk score of applications. The table reveals that Random forest makes predictions with an error rate of 0.059 while the K-Nearest Neighbors makes predictions with an
error rate of 0.2. Figure 5.4 presents the results represented in graphs.

![Bar chart showing Mean Absolute Error results of the models](image)

Figure 5.4: Mean absolute error results of the models

Table 5.7: Random Forest and K-NN Regression Results.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Absolute Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>0.0590403201327643</td>
</tr>
<tr>
<td>K-Nearest Neighbors</td>
<td>0.2</td>
</tr>
</tbody>
</table>
5.3 Distribution Analysis Results

The distribution analysis resulted in histograms and QQ plots. All graphs were studied and figure 5.5, 5.6, 5.7, and 5.8 show the distributions found.

Figure 5.5 follows the patterns of a right-skewed distribution with most of the data residing on the left side as shown in 5.5b.

(a) QQ Plot  (b) Histogram plot

Figure 5.5: Random Forest Regression Distribution Analysis

Figure 5.6 follows the patterns of a bimodal distribution with the data appearing to have multiple peaks.

(a) QQ Plot  (b) Histogram Plot

Figure 5.6: K-NN Regression Distribution Analysis

Figure 5.7 follows the patterns of a bimodal distribution with the data appearing to have multiple peaks.

Figure 5.8 follows the patterns of a bimodal distribution with the data appearing to have multiple peaks.
Chapter 5. Results and Analysis

(a) QQ Plot  
(b) Histogram Plot  

Figure 5.7: Random Forest Classification Distribution Analysis

(a) QQ Plot  
(b) Histogram Plot  

Figure 5.8: K-NN Classification Distribution Analysis

5.4 Mann-Whitney U test Results

The results of the distribution analysis demonstrate various distributions of the samples. The Mann-Whitney U-test which is a nonparametric test used to test a hypothesis was used due to the varying distribution of samples.

The results are summarized in Table 5.8. The results indicated the null hypothesis was not rejected when comparing the models’ performances in classification and regression.

Table 5.8: Summary of the Mann-Whitney U-test results.

<table>
<thead>
<tr>
<th>Model</th>
<th>Model</th>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>K-Nearest Neighbors</td>
<td>Regression</td>
<td>Fail to reject $H_0$</td>
</tr>
<tr>
<td>Random Forest</td>
<td>K-Nearest Neighbors</td>
<td>Classification</td>
<td>Fail to reject $H_0$</td>
</tr>
</tbody>
</table>

The details of the results when compared can be found in appendix C.
Chapter 6  
Discussion

The findings of this study identify static, dynamic, and grey-box analysis as methods for evaluating mobile application security. Control-flow analysis, data-flow analysis, lexical analysis, static taint analysis, dynamic taint analysis, behavioral analysis, regression analysis, pattern testing, matrix testing and machine learning are techniques that adopt these methods. Machine learning models found to be suitable for security risk assessment include random forest, k nearest neighbors, support vector machine (SVM), logistic regression, and decision trees. The findings suggest that the attributes on the application distribution platform uncover the likelihood and impact of a security threat and the results show that both K-Nearest Neighbors and Random forest models are effective in evaluating the security risk associated with installing mobile applications based on information available on the application distribution platform.

The distribution analysis was used to determine the type of test for the hypothesis. The QQ-plots and histograms showed various distributions that lead to the type of test selected. After performing the statistics test, there was no proof to establish a significance in the performance of the models in assessing security risk for mobile applications. However, the random forest model seemed to make fewer errors in predicting the risk score, but on the other hand, the k nearest neighbors model appeared to be slightly more accurate in predicting the security risk class of applications.

While previous research focused on evaluating how models perform in distinguishing malicious applications from benign ones, the findings in this study provide new insight to demonstrate the performance of various models in identifying the security risk of installing applications with the use of attributes provided on the applications distribution platform supporting the results of studies such as [10] and [9] that suggest these methods are effective and produce accurate results.

The method used in data collection limits the generalization of the data because the applications do not cover all the categories on the distribution platform. The permissions also do not represent all the permissions of the application and they are not obtained as a result of disassembling the applications, but transcription of the permission titles presented on the app distribution platform. The size of the dataset used to train the models was relatively small and this could also be a reason for the insignificance in the results as the models could make mistakes.

Therefore, further research is necessary to understand how a model will perform in evaluating the security risk of installing applications.
Chapter 7

Conclusions and Future Work

7.1 Conclusions

This research identifies various methods and techniques for assessing mobile application security risk, investigates how application attributes indicate the harmfulness of applications and evaluates the performance of Random Forest and K-Nearest Neighbors machine learning models in assessing the security risk of installing mobile applications based on information available on the app distribution platform. The results show that static, dynamic, and grey-box analysis are the methods used to evaluate mobile application security, and machine learning models including K-NN and Random forest are suitable techniques for evaluating mobile application security risk. Attributes such as the permissions, number of installations, and ratings reveal the likelihood and impact of an underline security threat. The K-NN and Random forest models when compared to evaluate the security risk of installing mobile applications based on information on the application distribution platform showed high performance with little differences.

RQ1 How should security risk be assessed for mobile applications?
Security risk assessment is the identification of security flaws in the software. Different methods exist for assessment and they have been generally categorized into Static, Dynamic, and Grey-box analysis.

RQ1.1 What machine learning techniques are suitable for evaluating mobile application security risk?
Studies have identified that K-Nearest Neighbors, Random forest, Decision trees, Support Vector Machine(SVM), and Logistic regression are suitable for evaluating mobile application security risk with high accuracy.

RQ2 How do the attributes (permissions, privacy policies, etc.) indicate the harmfulness of an application?
The attributes uncover details about the likelihood and impact of an underlying security threat. The impact of a threat can be described by attributes such as the Permissions while the likelihood of an attack can be evaluated by investigating the attributes such as the developer’s reputation that includes app’s reputation, size of the company, number of years on the market, frequency of updates, available privacy policy and compliance with standards, regulations, and ratings.

RQ3. How do different models based on application metadata perform when implemented to evaluate mobile application security risk?
As seen in table 5.6 and 5.7, The random forest model predicts the risk score and class of an application with an error rate of 0.059 and 82% accuracy respectively while the K-Nearest Neighbors predicts the risk score and class of an application with an error rate of 0.2 and an accuracy of 85% respectively.

7.2 Future Work

This study provides an analysis of machine learning models in evaluating the security risk of installing mobile applications based on information provided on the app distribution platform. However, it will be interesting to examine the performance of other models considering other application distribution platforms and other attributes such as the application description. It will also be beneficial to have a larger dataset.
Bibliography


[36] Tejas Vithani and Anima Anandkumar. “Modeling the Mobile Application Development Lifecycle”. In:


Appendix A

Expert evaluation of applications security risk score and class

Tables A.1, A.2, A.3, A.4 present the expert evaluation of the 60 mobile applications studied showing the risk factor, weight factor, security risk score and class of the applications as a result of the assessment.
## Table A.1: Results of the Expert Evaluation of applications.

<table>
<thead>
<tr>
<th>App Name</th>
<th>Rating</th>
<th>Offered By</th>
<th>P Policy</th>
<th>email</th>
<th>website</th>
<th># Installs</th>
<th>Permission</th>
<th>Security Risk</th>
<th>Class</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect Piano</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>31.0</td>
</tr>
<tr>
<td>Paint By Number - Free Coloring Book &amp; Puzzle ...</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>17.2</td>
</tr>
<tr>
<td>Akinator</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>39.1</td>
</tr>
<tr>
<td>SHEIN-Fashion Shopping Online</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>71.3</td>
</tr>
<tr>
<td>Love Messages for Girlfriend Flirty Love Let...</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>24.1</td>
</tr>
<tr>
<td>Amino: Communities and Chats</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>47.1</td>
</tr>
<tr>
<td>Curvy Singles Dating - Meet online, Chat &amp; Date</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>47.1</td>
</tr>
<tr>
<td>Sua Música</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>35.6</td>
</tr>
<tr>
<td>iQIYI Video – Dramas &amp; Movies</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>42.5</td>
</tr>
<tr>
<td>Amazon Music: Stream Trending Songs &amp; New Beats</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>17.2</td>
</tr>
<tr>
<td>AIMP</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>23.6</td>
</tr>
<tr>
<td>Bukalapak - Jual Beli Online</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>64.4</td>
</tr>
<tr>
<td>Hepsiburada</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>33.3</td>
</tr>
<tr>
<td>Tiki Shopping &amp; Fast Shipping</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>46.0</td>
</tr>
<tr>
<td>App Name</td>
<td>Rating</td>
<td>Offered By</td>
<td>P Policy</td>
<td>email</td>
<td>website</td>
<td># Installs</td>
<td>Permission</td>
<td>Security Risk</td>
<td>Class</td>
<td>Score</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>--------</td>
<td>------------</td>
<td>----------</td>
<td>-------</td>
<td>---------</td>
<td>------------</td>
<td>------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>eBay Kleinanzeigen for Germany</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>28.2</td>
<td>0.4</td>
<td>Medium</td>
</tr>
<tr>
<td>Trendyol - Hızlı ve Güvenli Alışverişin Yolu</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>10.3</td>
<td>0.4</td>
<td>Low</td>
</tr>
<tr>
<td>ROMEO - Gay Dating &amp; Chat</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>51.7</td>
<td>0.4</td>
<td>High</td>
</tr>
<tr>
<td>PinkCupid - Lesbian Dating App</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>21.3</td>
<td>0.4</td>
<td>Low</td>
</tr>
<tr>
<td>Elite Singles: Dating App for singles over 30</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>12.1</td>
<td>0.4</td>
<td>Low</td>
</tr>
<tr>
<td>Sverige Chat &amp; Dating</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>31.0</td>
<td>0.4</td>
<td>Medium</td>
</tr>
<tr>
<td>Bale</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>73.6</td>
<td>0.4</td>
<td>High</td>
</tr>
<tr>
<td>WIPPY - When you need a friend nextdoor</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>43.7</td>
<td>0.4</td>
<td>Medium</td>
</tr>
<tr>
<td>VK — live chatting &amp; free calls</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>80.5</td>
<td>0.4</td>
<td>High</td>
</tr>
<tr>
<td>Telegram X</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>52.3</td>
<td>0.4</td>
<td>High</td>
</tr>
<tr>
<td>Facebook Lite</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>62.1</td>
<td>0.4</td>
<td>High</td>
</tr>
<tr>
<td>All social media and social networks in one app</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1.0</td>
<td>40.2</td>
<td>0.4</td>
<td>Medium</td>
</tr>
<tr>
<td>Snapchat</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>55.2</td>
<td>0.4</td>
<td>High</td>
</tr>
<tr>
<td>TikTok</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>33.3</td>
<td>0.4</td>
<td>Medium</td>
</tr>
<tr>
<td>Mad For Dance - Taptap Dance</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>17.2</td>
<td>0.4</td>
<td>Low</td>
</tr>
</tbody>
</table>
Table A.3: Results of the Expert Evaluation of applications(3).

<table>
<thead>
<tr>
<th>App Name</th>
<th>Rating</th>
<th>Offered By</th>
<th>P Policy</th>
<th>email</th>
<th>website</th>
<th># Installs</th>
<th>Permission</th>
<th>Security Risk Class</th>
<th>Security Risk Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instagram</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>Medium</td>
</tr>
<tr>
<td>House Flip</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Scrabble® GO - New Word Game</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Moto Rider GO: Highway Traffic</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Bullet League - 2D Battle Royale</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Idle War: Legendary Heroes</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Real Boxing ñ€“ Fighting Game</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Dragon City - Collect, Evolve &amp; Build your Island</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Dice Dreams</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Property Brothers Home Design</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Snake.io - Fun Addicting Arcade Battle .io Games</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Cooking Fever</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Solitaire Showtime: Tri Peaks Solitaire Free &amp;...</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Free QR code Scanner app</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
<tr>
<td>Radyo Kulesi - Turkish Radios</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
<td>Low</td>
</tr>
</tbody>
</table>
Table A.4: Results of the Expert Evaluation of applications(4).

<table>
<thead>
<tr>
<th>App Name</th>
<th>Rating</th>
<th>Offered By</th>
<th>P Policy</th>
<th>email</th>
<th>website</th>
<th># Installs</th>
<th>Permission</th>
<th>Security Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RF</td>
<td>RF</td>
<td>WF</td>
<td>RF</td>
<td>WF</td>
<td>RF</td>
<td>WF</td>
<td></td>
</tr>
<tr>
<td>Smule - The Social Singing App</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>Medium 16.5</td>
</tr>
<tr>
<td>True ID Caller Name: Caller ID, Call Block</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 24.8</td>
</tr>
<tr>
<td>AliExpress - Smarter Shopping, Better Living</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>Medium 18.0</td>
</tr>
<tr>
<td>Kate Mobile for VK</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>1</td>
<td>0</td>
<td>Medium 14.3</td>
</tr>
<tr>
<td>OK</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>High 40.2</td>
</tr>
<tr>
<td>Samsung music</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>Medium 11.5</td>
</tr>
<tr>
<td>My Mixtapez Music</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 26.9</td>
</tr>
<tr>
<td>11st</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 23.2</td>
</tr>
<tr>
<td>Imo</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 27.2</td>
</tr>
<tr>
<td>Video chat - Oz Cam</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 22.6</td>
</tr>
<tr>
<td>Super Backup &amp; Restore</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 38.4</td>
</tr>
<tr>
<td>Nymgo: Cheap VoIP International Mobile, Call I...</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 21.3</td>
</tr>
<tr>
<td>MobileVOIP Cheap international Calls</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 24.1</td>
</tr>
<tr>
<td>DateU Pro - Meet, Love &amp; Date</td>
<td>1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>1</td>
<td>0</td>
<td>High 22.9</td>
</tr>
<tr>
<td>WAVE - Video Chat Playground</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 24.6</td>
</tr>
<tr>
<td>SMOOTHY - Group Video Chat</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>High 22.3</td>
</tr>
</tbody>
</table>
Appendix B

Machine Learning Models

Section B.1, B.2, B.3, and B.4 presents the code used to implement the models.

B.1 Random Forest Regression

The random forest model in performing regression takes application attributes as input variables and predicts the security risk score obtained as a result of the expert evaluation. The random forest model is developed using the code below

```python
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score, cross_val_predict

df = pd.read_csv('secassessment.csv')

# Convert non-numeric data to numeric
df.classification[df.classification == 'High'] = 3
df.classification[df.classification == 'Medium'] = 2
df.classification[df.classification == 'Low'] = 1

# define dependent variables (output)
Y = df['risk_score'].values

# Convert dependent variable type to int
Y = Y.astype('int')

# Define independent variables
X = df.drop(columns=['app_name', 'risk_score', 'classification'])

# Splitting the data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,

# Create the parameter grid based on the results of random search
# param_grid = {'bootstrap': [True],
```
B.1. Random Forest Regression

```python
# 'max_depth': [30, 50, 80, 100, 110],
# 'max_features': [2, 3],
# 'min_samples_leaf': [3, 4, 5],
# 'min_samples_split': [8, 10, 12],
# 'n_estimators': [10, 20, 30, 40, 50, 100]
#
# Random Forest classification
# ml = RandomForestRegressor()

# Instantiate grid search and fit the model
# grid_search = GridSearchCV(estimator=ml, param_grid=param_grid, cv=3,
# model = grid_search.fit(X_train, Y_train)
# print(grid_search.best_params_)

ml = RandomForestRegressor(n_estimators=10, max_depth=30, max_features=2,
 random_state=30)

model = ml.fit(X_train, Y_train)

# Predict mean absolute error
prediction_test = ml.predict(X_test)

# train_score = ml.score(X_train, Y_train)
# test_score = ml.score(X_test, Y_test)

errors = metrics.mean_absolute_error(Y_test, prediction_test)
# print('Mean Absolute Error', errors)

# Perform k-fold cross validation
scores = cross_val_score(model, X, Y, cv=10, scoring='neg_mean_absolute_error')
print('Cross Validation scores:', scores)

# Cross validation predictions
predictions = cross_val_predict(model, X, Y, cv=10)

accuracy = metrics.mean_absolute_error(Y, predictions)
print('CV MAE Acc:', accuracy)
```
B.2 Random Forest Classification

The random forest model in classifying applications takes application attributes as input variables and predicts the security risk class obtained as a result of the expert evaluation. The random forest model is developed using the code below:

```python
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sn

df = pd.read_csv('secassessment.csv')

# Convert non-numeric data to numeric
df['classification'][df['classification'] == 'High'] = 3
df['classification'][df['classification'] == 'Medium'] = 2
df['classification'][df['classification'] == 'Low'] = 1

# define dependent variables (output)
Y = df['classification'].values

# Convert dependent variable type to int
Y = Y.astype('int')

# Define independent variables
X = df.drop(columns=['app_name', 'risk_score', 'classification'])

# Splitting the data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
                                                  random_state=42)

# Random Forest classification
ml = RandomForestClassifier(n_estimators=10, max_depth=30, max_features=2)
model = ml.fit(X_train, Y_train)

# Predict mean absolute error
prediction_test = ml.predict(X_test)

# Perform k-fold cross validation
scores = cross_val_score(model, X, Y, cv=10)

# Perform k-fold cross validation
scores = cross_val_score(model, X, Y, cv=10, scoring='accuracy')
```
print("> Cross Validation scores:", scores)

# Cross validation predictions
predictions = cross_val_predict(model, X, Y, cv=10)

# Accuracy
accuracy = metrics.accuracy_score(Y, predictions)
print('> CV MAE Acc: ', accuracy)

# confusion matrix
cm = confusion_matrix(Y, predictions)
print(cm)

# Plot confusion matrix
df_cm = pd.DataFrame(cm, index=['Low', 'Medium', 'High'], columns=['Low',
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, annot_kws={'size':12})

plt.show()
B.3 K-NN Regression

The K-NN model in performing regression take the applications’ attributes as input variables and predicts the security risk score of the application obtained as a result of the expert evaluation. The KNN model is developed using the code below

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn import metrics
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import GridSearchCV

df = pd.read_csv('secassessment.csv')

# Convert non-numeric data to numeric
df['classification'][df['classification'] == 'High'] = 3
df['classification'][df['classification'] == 'Medium'] = 2
df['classification'][df['classification'] == 'Low'] = 1

# Define independent variables
X = df.drop(columns=['classification', 'app_name', 'risk_score'])

# define dependent variables (output)
Y = df['classification'].values

# Convert dependent variable type to int
Y = Y.astype('int')

# Splitting the data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,

# k-Nearest Neighbours
clf = KNeighborsRegressor(n_neighbors=1)

# Fit model to the data
model = clf.fit(X_train, Y_train)

# Test the model in classification
predictions = clf.predict(X_test)
```
B.3. K-NN Regression

```python
errors = metrics.mean_absolute_error(Y_test, predictions)
print('CV MAE Acc: ', errors)

# k-fold cross validation
knn_cv = KNeighborsRegressor(n_neighbors=1)

# train model with cv
cross_score = cross_val_score(knn_cv, X, Y, cv=10, scoring='neg_mean_absolute_error')
print('CV Scores: ', cross_score)

# Cross validation predictions
predictions = cross_val_predict(model, X, Y, cv=5)
accuracy = metrics.mean_absolute_error(Y, predictions)
print('MAE :{} '.format(accuracy))

# Hypertuning model using GridSearchCV
# knn2 = KNeighborsRegressor()
#
# Create a dictionary of all values to test for n_neighbors
# param_grid = {'n_neighbors': np.arange(1, 25)}
#
# Test all values
# knn_gscv = GridSearchCV(knn2, param_grid, cv=10)
#
# fit model
# knn_gscv.fit(X, Y)
#
# Check value of best performing n_neighbors
# print('GridSearch CV neighbor', knn_gscv.best_params_)
#
# check mean score for top performing value
# print('Accuracy using GridSearch CV neighbor', knn_gscv.best_score_)
```
B.4 K-NN Classification

The K-NN model in classifying applications take the application attributes as input variables and predicts the security risk class obtained as a result of the expert evaluation. The KNN model is developed using the code below:

```python
import pandas as pd
import numpy as np
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sn

df = pd.read_csv('secassessment.csv')

# Convert non-numeric data to numeric
df['classification'] = ['High', 'Medium', 'Low']

# Define independent variables
X = df.drop(columns=['classification', 'app_name', 'risk_score'])

# define dependent variables (output)
Y = df['classification'].values

# Convert dependent variable type to int
Y = Y.astype('int')

# Splitting the data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,

# k-Nearest Neighbours
clf = KNeighborsClassifier(n_neighbors=1)

# Fit model to the data
model = clf.fit(X_train, Y_train)

# Test the model in classification
predictions = clf.predict(X_test)
```
#check accuracy of the model
acc = clf.score(X_test, Y_test)
print(’> k-NN ACCURACY : ’, acc)

# k-fold cross validation
knn_cv = KNeighborsClassifier(n_neighbors=1)

cross_score = cross_val_score(knn_cv, X, Y, cv=10)
print(’> CV Scores: ’, cross_score)
cvpredictions = cross_val_predict(knn_cv, X, Y, cv=10)
accuracy = metrics.accuracy_score(Y, cvpredictions)
print(’> CV Accuracy: ’, accuracy)

cm = confusion_matrix(Y, cvpredictions)
print(cm)

# Hypertuning model using GridSearchCV
# knn2 = KNeighborsClassifier()
#
# # Create a dictionary of all values to test for n_neighbors
# param_grid = {’n_neighbors’: np.arange(1, 25)}
# # Test all values
# knn_gscv = GridSearchCV(knn2, param_grid, cv=10)
# # fit model
# knn_gscv.fit(X, Y)
# # Check value of best performing n_neighbors
# print(’> GridSearch CV neighbor’, knn_gscv.best_params_)
# # check mean score for top performing value
# print(’> Accuracy using GridSearch CV neighbor’, knn_gscv.best_score_)

# confusion matrix plot
df_cm = pd.DataFrame(cm, index=['Low', 'Medium', 'High'], columns=['Low', 'Medium', 'High'])
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, annot_kws={'size':12})
plt.show()
Appendix C

Mann-Whitney U Test

In table C.1 it can be seen that the null hypothesis was not rejected when comparing the Regressions and Classification results the models. The results suggest that there is no evidence to support if the difference in the samples are significant.

Table C.1: Results of Regression and Classification Mann-Whitney U Test

<table>
<thead>
<tr>
<th>Model</th>
<th>Operation</th>
<th>N₁</th>
<th>N₂</th>
<th>R₁</th>
<th>R₂</th>
<th>U₁</th>
<th>U₂</th>
<th>p(two-sided)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>Regression</td>
<td>10</td>
<td>10</td>
<td>95</td>
<td>115</td>
<td>60</td>
<td>40</td>
<td>p &gt; α (fail to reject H₀)</td>
</tr>
<tr>
<td>Random Forest</td>
<td>Classification</td>
<td>10</td>
<td>10</td>
<td>70</td>
<td>120</td>
<td>85</td>
<td>35</td>
<td>p &gt; α (fail to reject H₀)</td>
</tr>
</tbody>
</table>