الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية العلوم

قسم: الإعلام الآلي

Mémoire de Master

Spécialité : *Computer Security and Cryptography*

Thème

Android malware detection: Feature Extraction Issue

Presented by :

- Aouissi Mohamed Elamine
- Boufenik Omrane

Supervior :

- Dr. Mebarka Yahlali

Promotion 2023 - 2024

Dedication

I would like to dedicate this work to my loving family, whose unwavering support and encouragement have been my guiding light throughout this journey, Your belief in me has been my greatest strength, and I am forever grateful for your love, patience, and understanding.

To my parents, thank you for your endless sacrifices and for always believing in my dreams, Your unconditional love and encouragement have shaped me into the person I am today.

To my siblings, thank you for your constant support and for being my pillars of strength, Your presence in my life brings me joy and inspiration every day.

To all my friends, Especially my colleague **BOUFENIK OMRANE** thank you for your friendship, and support, Your encouragement has kept me going during the toughest times.

To my best friend **Djab Allah Mokeddem** whose unwavering support have been my guiding light, Your friendship is a constant source of strength and inspiration. Finally,This work is dedicated to each and every one of you.

With love and gratitude,

AOUISSI MOHAMED ELAMINE

Dedication

I dedicate this work to my family, especially to my father and mother who were the reason for who I am today. I dedicate this dissertation to all my teachers, my colleagues, and my university family. Without forgetting my partner amine and all my best friends.

To my cherished friend Djab Allah, Thank you, for your constant support and encouragement.

BOUFENIK OMRANE

Acknowledgements

First and foremost, I am immensely thankful to God for His blessings, guidance, and grace throughout this journey, Without His divine assistance, none of this would have been possible.

Second, I am immensely thankful to my supervisor, **Dr.Mebarka Yahlali**, for her invaluable guidance, support, and encouragement throughout this journey, Her expertise and wisdom have been instrumental in shaping this research and pushing me to strive for excellence.

I would like to express my sincere gratitude to the professors who generously gave their time and expertise to review and give feedback on my thesis, I greatly appreciate them for their valuable guidance, and contributions which will be helpful in shaping the quality of my research.

I am deeply grateful to all my family, friends, and colleagues who have supported and encouraged me along the way, Their unwavering belief in me has been a constant source of inspiration.

Lastly, I would like to thank all the participants who generously contributed their time and knowledge to this research, Your involvement has been invaluable, and I am truly thankful for your contributions.

With sincere thanks and gratitude.

Contents

List of Figures

List of Abbreviations

API Application Programming Interface. 7

APK Android Application Package. 2

AVG Anti-Virus Guard. 16

DAC Discretionary Access Control. 10

DR Dimensionality Reduction. 36

GID Group ID. 10

ICA Independent Component Analysis. 32

IDS Intrusion Setection System. 16

IG Information Gain. 22

IM Information Mutuelle. 64

iOS iPhone Operating System. 5

ISOMAP Isometric mapping. 34

KPCA Kernal PCA. 33

LDA Linear Discriminant Analysis. 31

OS Operating System. 2

PCA Principal Component Analysis. 24

SMS Short Message Service. 7

SVD Singular Value Decomposition. 30

t-SNE t-Distributed Stochastic Neighbor Embedding. 34

UID User ID. 10

XNU X is Not Unix. 5

General Introduction

Nowadays, with the advancement of technology, computers have been replaced by more portable devices like smart wristbands, smart mobile devices, tablets, ect.
The Android OS is one of the most popular operating systems used on these devices, users can easily download serval apps on their mobile devices via the Android App Store.
Certainly, mobile devices make life easier but these connected devices are constantly collecting our data (usage, location, communications and more), users are often unaware of how much and what type of data is collected, And if this data got into the wrong hand, Our identity and privacy would be stripped away from us.
Malware developers try to access to the personal information through these apps, They can access user's devices by injecting malware into an APK file that represents an extension of Android-based applications.
Several works have been developed in the field of machine learning for android malware detection, the first work related to Android security is therefore focused on analyzing the security limits under Android and on a way to overcome them, However, this type of approach has one main limitation: it cannot detect and learn new attacks, other work is based on information flows [**1**], where it is necessary to learn how attacks take place by directly analyzing malicious applications and using the knowledge base acquired during learning to detect these malwares, the complexity of a classification algorithm has grown significantly these last decade when the mass of data (number of samples in the database as well as the size of **their description**) has greatly increased.
Current technology, which enables real-time analytics to allow faster and more responsive decision-making, produces a strong need to process and analyze huge datasets in a real-time manner, the complexity of an algorithm becomes the key concern to reduce the dimension of the data, Consequently, feature selection or features extraction could affect the quality of machine learning model.
Objective : The objective of this work is to illustrate the importance of dimension reduction of the data set by extraction in the case of Android malware detection data sets.
We have tested and compared several approaches on several data sets and finally we have proposed a new approach based on the studied approaches.
Organization: The work is organized as follows:
ChapterI: Mobile Application Security This chapter presents the security of mobile applications and the Android security model and its features, as well as

Android app permissions, malware detection works and techniques on Android
ChapterII: Features extraction This chapter introduce dimension reduction problem.
We present the different features extraction methods proposed in the literature.
ChapterIII: Contribution and Implementation Chapter III explains the experiment settings, such as the chosen datasets, the classification model, the evaluation over the model after the applying feature extraction algorithm and finally the proposed approach.

Chapter I

Mobile By Android

I.1 Introduction

Malware keeps increasing for mobile apps Without a doubt, as a result, researchers are spending significant resources to improve malware detection techniques, in order to understand the behavior of these malicious software and analyze those of Android, it is necessary to take an interest in the functioning of the OS itself, in this chapter, we first present mobile applications with a focus on the most popular mobile operating systems, then, we continue with the Android operating system by specifying its architecture, as well as the development strategies for Android applications, finally, we present the security of this system.

I.2 Mobile applications

Mobile apps consist of a set of programs that run on a mobile device and perform certain tasks for the user, the mobile app can be used in most mobile devices, including inexpensive and basic mobile devices, the mobile app has multiple uses for its vast field of operation, such as calling, sending messages, browsing, chatting, communicating on social networks, audio, video, etc. [2]

I.2.1 Mobile Operating Systems

Just like a computer has an Operating System (OS), mobile devices also have an Operating System known as mobile OS, mobile OS is the software that provides an environment in which the user of the mobile device runs application programs conveniently and efficiently[3].

- **IOS**

  iPhone Operating System (iOS) is a mobile OS developed by Apple, it was originally called iPhone OS, but was renamed iOS in June 2009, iOS currently runs on iPhone, iPod touch and iPad, which they share the foundations: an XNU kernel based on the Mach micro-kernel, the various Unix and Cocoa services, etc. iOS has 4 layers of abstractions similar to MacOS [4]:

  ◇ A layer < OS kernel >.

  ◇ A layer < Services kernel >.

  ◇ A layer < Media >.

  ◇ A layer < Cocoa >.

Figure I.1: iOS [4]

- **Android**

  Launched in 2005 by the start-up of the same name and then bought by Google in 2007, Android is a mobile OS based on a Linux kernel, it is considered a 'software stack' that behaves like an operating system [4].

Figure I.2: Android (OS) [4]

I.3 Android System

I.3.1 Android System Architecture

The Android OS is a stack of software components that can be divided into four layers (Figure I.3), each layer performs a specific set of tasks and communicates to the other layers via clearly defined interfaces.

1. **Kernel layer:** provides basic system features such as process management, memory management, device management including camera, display, keyboard, etc, The reason for choosing the Linux kernel for Android OS is that Linux is really good for basic operations such as networking. [5]

2. **Native Library Layer:** At the top of the Linux kernel layer are the native Android libraries, this layer allows the device to handle different types of data that are hardware specific, it has two essential parts; one is the Android library and the second is the Android runtime, all these libraires are written in C++ programming language. [5]

3. **Application Framework Layer:** The application structure layer is above the native library layer, the application layer provides a major application programming interface (API) and higher-level services in the form of java classes. Application developers are allowed access to all API structures for core programs that simplify the reuse of these components, these APIs are open to everyone to create Android apps. [5]

4. **Application Layer:** In Android architecture, the application layer is the top layer, these include both native application to pre-installed with each device, such as: SMS client, web browser contact manager, etc. An average Android device user would primarily interact with this layer for basic functions such as making phone calls, sending text messages, capturing images, browsing the web, playing videos and audios. [5]

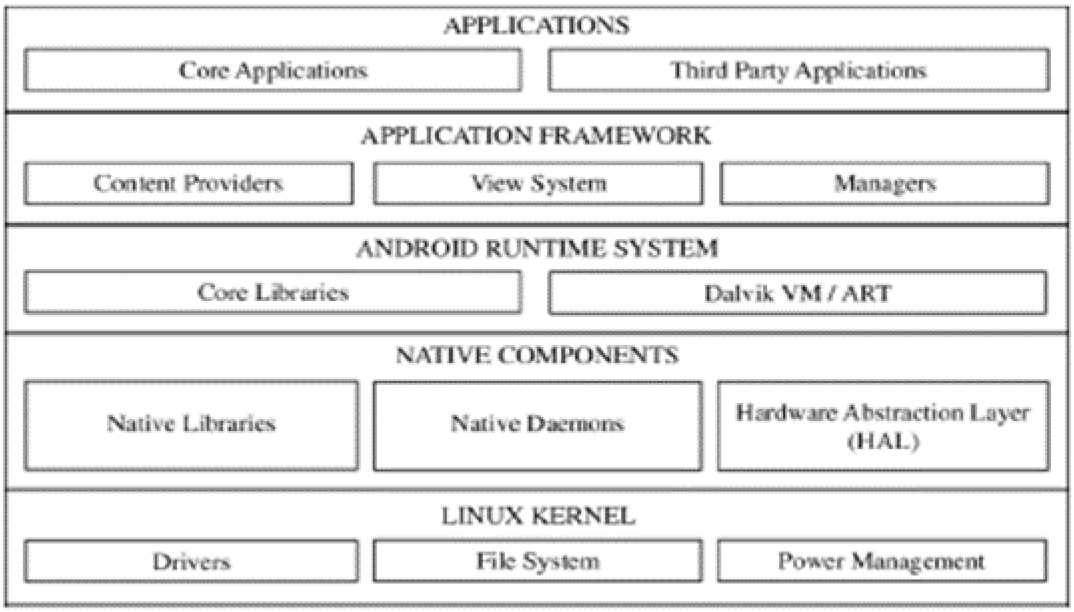


Figure I.3: The architecture of the Android OS [5]

I.3.2 The Android apps

An Android app is specifically developed for mobile devices using the Android system, they are very variable in nature such as games, mobile commerce, utility, information service [6], to understand how an Android app works we will present its architecture, communication between components (Intents), Intent-filter and Android packages, as well as AndroidManifest.XML.

- Architecture of an Android application
  An Android application can contain several components, each of them can be an input point into the program, there are four types of components that carry information through messages called Intents [1].

1. **Activity:**
   is one of the building blocks of Android OS, simply put, activity is a screen with which the user interacts, every activity in Android has a life cycle as created, started, resumed, paused, stopped or destroyed, these different states are known as Activity Life cycle. [1]
2. **ContentProvider:**
   used to share data of an application, which serves as an interface between the application wishing to access the data. They are stored in a local SQLite database but no restrictions are imposed on how to store this data. [1]
3. **Service:**
   performs tasks in the background, it is used to run long internal tasks or to run a task at the request of an application. [1]
4. **BroadcastReceiver:**
   used to listen to messages in wide distribution on the system, when a new SMS is received by the mobile device, the system sends a broadcast message to notify the different SMS sending and receiving applications. [1]

◇ **Intents:** the intent ensure and facilitate interaction between Android components. An Intent is an object used to start an operation or send a data set to another component, an Intent is a message with an action request and optionally, data made to another. [1]

◇ **Intent-Filter:** Structured description of the intent values to be matched, an Intent-Filter can match actions, categories, and data (via its type, schema, and/or path) in an Intent, it also includes a ”priority” value used to order multiple matching filters. [6]

```xml
<activity android:name=".MainActivity">
    <!-- This activity is the main entry, should appear in app launcher -->
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
<activity android:name=".ResultActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Figure I.4: Exemple Intent-Filter dans Manifest Android [6].

◇ **Android Package:** The objects contain version information about the implementation and specification of a Java package, this version information is retrieved and made available by the Loaderinstance class. [7]

⋆ **AndroidManifest.xml:** The AndroidManifest.xml file describes the essential information in the application (APK file), some important parts that can be mentioned are [8]: Package name, Application Components, Permissions manifestes, figure I.5 shows an example of the AndroidManifest.xml file of an Android Studio[1] application .

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mrcherif.nmap_auto">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".ThreadMain">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity" />
        <activity android:name=".hidtroy" />
    </application>

</manifest>
```

Figure I.5: File AndroidManifest.xml.[6]

[1]Development environment to develop Android mobile applications.

I.4 Android System Security

Android natively implements some mechanisms that offer a certain level of security.

1. **Permissions**
   The purpose of a permission is to protect the privacy of an Android user, Android apps must have permission to access to user sensitive data (such as contacts and calls), as well as certain system features (such as camera and internet), depending on the functionality, the system can automatically grant permission or can prompt the user to approve the request [9]. Each permission corresponds to a Linux kernel GID[2], each GID has access to the OS resources required to execute the behaviors associated with this permission, for each permission granted, Android adds an app UID (User ID) to the corresponding group, the application obtains the privileges to act with the requested permission [9].

Type	Description
Normal	The default for permissions. It is Automatically granted to any application requesting it
Dangereuse	The user must accept permission in order to grant it. Example READ-SMS for SMS access.
Signature	Permission granted only if the requesting application is signed with the certificate of the developer who declared the permission.
Signatureorsystem	Permission granted only to system applications, specifically those in the system partition, or those that were signed with the same signature as the application that declared the permission.

Figure I.6: Android permission types [1].

2. **User Unique Identifier (UID)**
   Role-based input control is introduced as a user ID (UID), by assigning one UID per Android application at the time of installation and forcing them to run only through this UID, each application is stored in a separate file space from other applications [9].

3. **Discretionary Access Control (DAC) and sandboxing**
   The DAC mechanism allows user access control to files and directories, it works in an invisible way for app developers and users. It separates

[2]Group ID: is used to manage multiple users in a regular Linux system

applications from system resources, in effect, it is used to allow or not applications to access system resources [10].

4. **Administration of the Android device**
   Android offers an API that enables the development of applications to administer mobile devices, the API allows to increase the security policy on passwords (exp: size, expiration and number of times has re-entered the password), impose encryption of partitions (enable/ disable WI-FI, Bluetooth, etc.), request the creation of a new password, lock the mobile device, etc. [1]

I.5 Limitations of Android Security Mechanisms

1. **Abuse of permission**
   Permissions give applications access to sensitive mobile device resources , if the user wants to install an application, he must grant it all requested permissions, if they filter access to these resources, there is no use verification of these resources, only application developers can ensure that there will be no abuse. Simple attacks also use permissions incorrectly, and this is the case for malware that aims to leak sensitive data from the mobile device [1].

2. **Permissions:**
   delegation and collusion attacks A delegation attack consists of delegating the execution of a task requiring permission that the malicious application does not have to another application, for example, an application that does not have permission to communicate on the network could use the browser to input information or download files, a collusion attack is a cooperation between several applications to lead an attack [1].

3. **Software vulnerabilities:**
   privilege elevation Like all current systems; Android system also has software vulnerabilities, exploiting some of them increases the privileges of an application and performs sensitive operations related to personal information.[1]

I.6 Android malware

In its different form, malicious applications represent a major problem affecting the Android operating system, This section describes the malware lifecycle, its family and the detection techniques.

I.6.1 Definitions

- **A malware:** a malware is a program or code with the main purpose of harming a given system.

- **A malware sample:** malware sample is an Android application that contains this malware, to analyze a malware is also to analyze one or more of its samples to extract the information related to this malware, to detect a malware is to decide if a given application is a sample of a malware [1].

I.6.2 Malware Life cycle

Malware for mobile platforms in general and Android in particular replicates the behavior of viruses encountered on desktop computers, their life cycle is structured around seven main phases [11]:

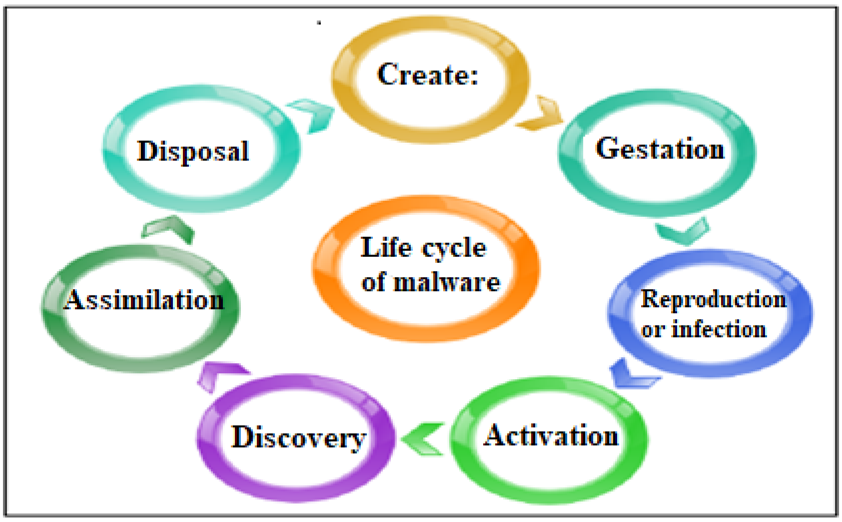


Figure I.7: Life cycle of malware.

1. **Create:** step in which the programmer designs and implements all the malicious code that will be included in the malware.

2. **Gestation:** step in which the malicious application infiltrates and installs into the system. It stays inactive throughout this step, that is why its presence remains completely unknown to the user.

3. **Reproduction or infection:** the malware reproduces itself a significant number of times before manifesting in this phase. The malware author seeks to remotely control devices and access to private data. Malware spreads through file sharing or social engineering techniques on Android. It uses SMS, Bluetooth, Wifi as a means of communication and often disguises itself as normal application.

4. **Activation:** some malware activates its destruction routine when certain conditions are met (the internal countdown reaches for example). Activation can also be done remotely, the goal of this phase is to gradually appropriate all device resources.

5. **Discovery:** the user notices strange behavior and suspects the presence of a malicious application, this strange behavior may include performance loss, changes in the web browser home page, or unavailability of some system functions.

6. **Assimilation:** antivirus software updates its virus database after the discovery of new malware. If possible, a solution or antidote is also proposed to eliminate this threat.

7. **Disposal:** this is the phase where the antivirus discovers the malware, prompts the user to remove it, it marks the death of the malware.

I.6.3 Malware Family

There are millions of different malwares. These malwares have different features, they can be classified by family [12]:

- **Backdoors:** allows the execution of remotely controlled operations that damage the device.

- **Commercial Spyware:** sends sensitive information without users' authorization such as tracking information.

- **Data collection:** extracts information about installed applications, user accounts or files from the device without user permission.

- **Downloader hostile:** downloads other harmful applications, although it does not include any code.

- **SMS Fraud:** provides interfaces that look like reliable sources, it uses these interfaces to request authentication or billing information that allows the user to send it to a third party.

- **Ransomware:** fully or partially controls the mobile or mobile data by locking the device or encrypting the data in order to demand the ransom to remove the control.

- **Spam:** delivers unwanted commercial messages to the user's contacts.

- **Spyware:** steals contacts, images, files, email content, call logs, message logs and browser history. In addition, recording phone or audio calls.
- **Trojan:** this type of application appears as a benign application because it hides its harmful actions against the user.

I.6.4 Malware Detection Techniques

Various techniques are focused on detecting Android malware, we will present above the detection techniques of the latter and these tools.

Static analysis:

Static analysis filters out parts of the application without actually running them, this technique integrates analysis based on signatures, permissions, and components. The signature-based strategy draws features and creates distinctive signs to identify specific malware, therefore, it is not enough to recognize the variation or unidentified malware. The permissions-based policy recognizes permission requests to distinguish malware. Component-based techniques decompile the application to draw and inspect the byte code definition and connections of important components (i.e., activities, services, etc.) to identify exposures, the main drawbacks of static analysis are the lack of actual execution paths and appropriate execution conditions, in addition, there are problems with the occurrence of code obfuscation and dynamic code loading. [13]

Static analysis tools:

Among the static analysis tools for applications under Android [14]:

- **Androguard:** is a framework that analyzes Android applications.
- IDA pro version 6.1: is a disassemble, a software used to translate machine code into a readable format.
- **APKInspector:** graphical interface tool to analyze an Android application.
- **Dex2jar:** A tool designed to perform the work of converting an Android application in dex format to a file of Java class format.
- **Jd-gui:** is a standalone graphical utility that displays the Java source codes of <.class> files. (Java).
- **JAD:** Java decompiler.
- **Dexdump:** decompiles JAVA files in DEX[3] format.

[3]Used to run applications developed for Android OS

- **Smali:** assembler/ disassembler for the DEX format used by dalvik[4].

Dynamic Analysis:
Dynamic analysis technique includes running the application on a virtual machine or physical device, in the middle of the exam, the behavior of the application is monitored and can be dissected, dynamic analysis gives a less abstract application perspective than static analysis, code paths executed during execution are a subset of each unique accessible path, the main objective of the analysis is to achieve high code inclusion, because every possible event must be enabled to monitor any possible malicious behavior, the main disadvantages of dynamic analysis are that dynamic analysis requires considerable resources compared to static analysis, which prevents it from being distributed on mobile devices with limited resources. In addition, dynamic analysis is responsible for low-code coverage. Recently, the malware has tried to recognize the emulator and other dynamic analysis frameworks and refrain from exposing their payloads [13].
Dynamic analysis tools:

There are a number of tools to perform dynamic analysis of an Android application, these are used to test malicious applications in a protected environment [14].

- **Droidbox:** tool for Android Sandbox[5] applications.
- **Mobile Sandbox:** tool for mobile applications that are available online.

Hybrid Analysis:
The hybrid analysis technique includes the consolidation of static and dynamic features collected during application review and data drawing while the application is running, Nevertheless, this would increase the accuracy of the identification, the main disadvantage of hybrid scanning, it consumes the resources of the Android system and takes a long time to perform the scan [13].

I.7 Countermeasures

Security is primordial and systems must be put in place to prevent all external and internal threats, this section describes an overview of the countermeasures that can be applied to reduce the risk of attacks against the Android system, among them:

[4]Is a virtual machine that executes files in Dalvik Executable(.dex) format

[5]Sand box: is a security feature that prevents Access from executing certain potentially dangerous expressions, these insecure expressions are blocked, so that the database is «reliable» (its content is enabled)

- **Antivirus:** antivirus is security software mainly used on mobile devices. The popularity gained on computers has contributed to increase the level of confidence gained by mobile users, Avast, AVG and F-Secure are examples of renowned antivirus in Android, they face new constraints brought about by the rapid evolution of malicious applications, like desktop platforms, their effectiveness is closely linked to their detection methods [11].

- **Firewall:** using a firewall on Android mobile devices may not be as critical as a PC, but it can be useful to manage Internet access for better security, data optimization and performance [15].

- **Intrusion detection:** an intrusion detection system (IDS) is a detection mechanism to discover attempts to compromise a system. Potentially, this can prevent such attempts, in this case, the system is called intrusion prevention system, intrusion detection mechanisms applied to Android mobile devices are based on the same principles as mechanisms used in other systems (personal computers and computer networks). although systems differ considerably in type and architecture, the foundations of attack protection remain the same. this allows the adoption of techniques and their use in the Android security zone [16].

- **Malware analysis:** Malware analysis deals with the study of how malware works and possible results of infection with a given specific malware. it is important to know that malware can have different features, each malicious feature is designed by attackers to enter the system through different sources to infect without user consent [17].

I.8 Conclusion

In this chapter, we presented an overview of mobile applications in general, and particularly the Android system, where we explained how this platform works and its architecture. We were then interested in the security mechanisms proposed by this system that provide a certain level of security. Then we defined Android malware, its lifecycle, its family and these detection techniques, finally, we concluded with the countermeasures that are applied to reduce the risk of attacks against this system, in the next chapter, we will study about dimensionality reduction and its usefulness in protecting the Android system.

Chapter II

Features extraction

II.1 Introduction

Automatic data classification is an important concept that is part of the data analysis process, the complexity of the classifier has significantly increased when the description of data has greatly augmented.
The reduction of dimensions is one of the most classic solutions of this problem, its objective is to select or extract an optimal subset of relevant characteristics according to a predefined criterion, this step makes the whole data more representative and improves the performance of the detection algorithm (classifier). In this chapter, we are interested in the dimension reduction, within the framework of the automatic classification for a decision-making purpose.

II.2 Dimension reduction

Dimension reduction is an essential step in the data preprocessing process (filtering, cleaning, etc.), indeed, for data belonging to a high-dimensional space, certain attributes provide no information or even express noise, others are redundant or correlated.
This makes decision algorithms complex, inefficient, less generalizable and difficult to interpret. Methods for reducing the dimension of representation space can be divided into attribute extraction methods and attribute selection methods.
In feature selection, information can be lost since some features should be excluded, however, in feature extraction, the dimension can be decreased without losing much initial feature dataset [18]
The main objectives of dimension reduction are:

- Identification of relevant attributes
- Reduce the size of data
- Prediction accuracy improvement
- Reduction of necessary storage space
- Avoiding overfitting
- Reduce executing and training time

II.2.1 Feature Selection

Attribute selection methods offer to choose a subset of attributes from the original set, the latter contains the essential information to represent the objects, in machine learning, the goal is to choose a subset of attributes to categorize/group objects, the attribute selection approach is preferred in areas where the original (unprocessed) attributes are required to maintain the physical properties of attributes [19]

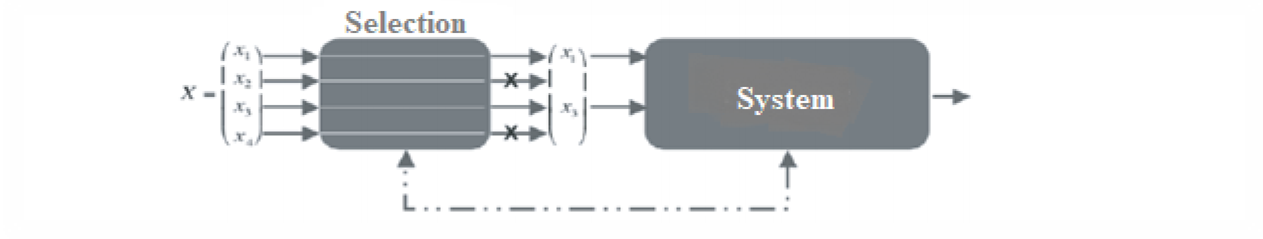


Figure II.1: Attribute Selection Process Extracted
[20]

II.2.2 Feature Extraction

The main idea is to transform the initial set of attributes into a new set, this new set of attributes better maintains the original information, if the extraction process produces a set of attributes larger than the original, the method is called attribute generation[19].

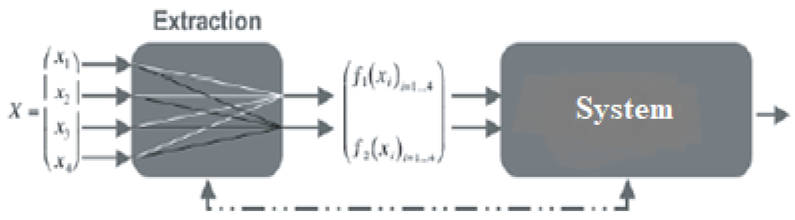


Figure II.2: Extract Attributes Process
[20]

In the machine learning community, these dimensioning methods can also fall into two categories: methods for supervised[1] learning and other for unsupervised[2] learning.
In recent decades, the study of methods for extracting attributes in general has

[1]Supervised: supervised learning aims to categorize objects in classes where the number of classes is known a priori

[2]Unsupervised: The goal of unsupervised learning is to group objects into clusters according to their similarity (the number of classes is unknown)

progressed enormously [21], the approach of attribute selection in the case of supervised learning is also well studied [22].

II.3 Feature Selection

Feature selection is used to reduce the dimensionality impact on the dataset through finding the subset of features which efficiently defines the data [23], it selects the important and relevant features to the mining task from the input data and removes redundant and irrelevant features [24], it is useful for detecting a good subset of features that is appropriate for the given problem [24], the main objective of feature selection is to construct a subset of features as small as possible but represents all vital characteristics of the input data [26].
Feature selection algorithm phase is divided into two-phase:(1) Subset Generation and (2) Subset Evaluation, in subset generation, we need to generate subset from the input dataset and in subset evaluations we have to check whether the generated subset is optimal or not [27]. **"FigureII.3"** shows the feature selection process

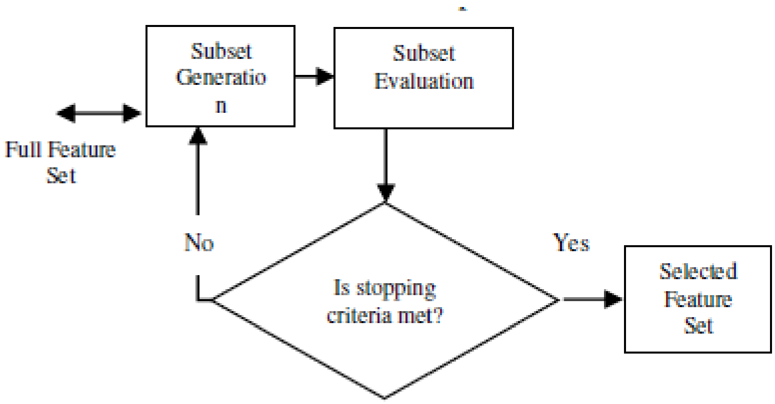


Figure II.3: Feature selection process
[27]

II.3.1 Categorization of attributes of selection methods

Feature selection methods are classified into three classes:

1. **Filters Methods**

   evaluate features without calling any classification algorithm, filter models uses statistical properties of variables to remove the variables that are not informative, these models can be 'Univariate' or 'multivariate', in the Univariate scheme [26] each feature is ranked independently of feature space while the multivariate scheme evaluates features in batch, filter models are easily scalable to very high dimensional datasets, computationally simple and fast, the cons of filter models are that they totally ignore the effect of selected feature subset on performance of induction algorithm [28].

Pros:

- It works faster than wrapper
- Scalable
- Classifier independent

Cons:

- The interaction between classifiers is neglected
- The dependency among the features is ignored

2. **Wrappers Methods**

use a predetermined learning algorithm to evaluate the quality of selected features and offer a simple and powerful way to address the problem of feature selection, the accuracy measured by this algorithm is very high as this method considers the interaction between feature subset searches and model selection, this method is more demanding than filter methods. [29]

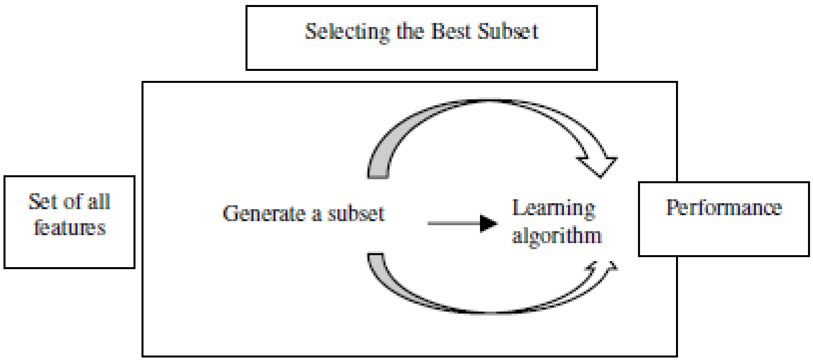


Figure II.4: Wrapper methods for feature selection [30]

Pros:

- Interacts with classifier
- Consider the dependence among features
- Higher performance accuracy than filter

Cons:

- Classifier specific
- Need expensive computation

3. **Hybrid models**

   are combination of both filter models and wrapper models, they include the features this two models, they are less computationally intensive and they include the interaction with model construction.[31]

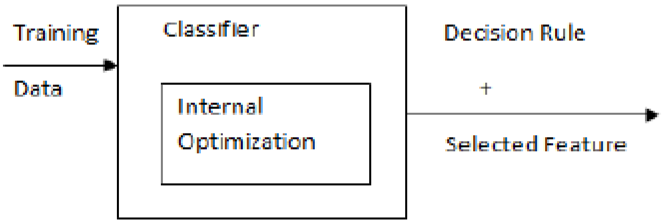


Figure II.5: Hybrid Model for Feature Selection
[**31**]

Pros:

- The performance accuracy is higher than filter
- Better computational complexity than wrapper

Cons:

- Classifier specific

II.3.2 Feature-Selection Methods

1. **Information Gain (IG):** One of the most commonly used univariate methods for evaluating attributes is the IG filter, it assesses features based on the information gained and examines each feature individually. The Information Gain filter employs a symmetrical measure, it sorts all features in a methodical manner and necessitates the establishment of a threshold for selecting a specific number of features based on the obtained order. [32].

2. **Chi-squared:** The Chi-squared test for feature selection is a statistical technique used to identify the most relevant features for a given set of data for a target variable, it works by comparing the observed distribution of the values of a characteristic with the expected distribution under the assumption of independence between the characteristic and the target variable and selecting those characteristics for which the difference between the observed and expected distributions is the largest [33].

3. **Relief:** Relief is a feature-selection method that serves as an individual evaluation filter. It computes a proxy statistic for each feature, which can estimate its quality or relevance to the target concept, these statistics are known as feature weights, or informally, as feature scores[34].

4. **ANOVA:** ANOVA is a widely recognized statistical method used for comparing multiple independent means, this technique evaluates features by computing the ratio of variances between and within groups and then ranks them accordingly [35].

5. **Symmetric Uncertainty:** Symmetric uncertainty is a means of determining the fitness of features for selection, it involves computing the relationship between the feature and the target class. [36].

6. **Recursive Feature Elimination (RFE):** Recursive feature elimination is a recursive greedy optimization approach, where features are selected by recursively taking a smaller and smaller subset of features, Now, an estimator is trained with each set of features, and the importance of each feature is determined using coef attribute [37].

II.3.3 Advantages of selecting features

There are various advantages of feature selection process: [38]

- Improved accuracy
- Simple models are easier to interpret.
- Shorter training times
- reduce Overfitting
- Easier to implement by software developers

II.4 Feature Extraction

Feature extraction aims to compress the data with the goal of maintaining most of the relevant information, Feature extraction is an important component of classification system, A well-defined feature extraction algorithm makes the classification process more effective and efficient.

Figure II.6: The process of feature extraction
[39]

II.4.1 Role of attribute extraction

Feature extraction allows machine learning models to improve their performance by [40]:

- **Reduces redundant data:** feature extraction cuts noise, removing redundant and unnecessary data. This enables machine learning programs to focus on the most relevant data.
- **Improves model accuracy:** the accuracy of machine learning models is improved when they use only the data required to train the model to its intended business use. The inclusion of peripheral data negatively affects the model's accuracy.
- **Accelerates learning:**The inclusion of training data that does not directly contribute to solving the business problem decelerate the learning process, models trained on highly relevant data learn faster and make more accurate predictions.
- **More efficient use of computing resources:** Trimming out peripheral data increases speed and efficiency, with less data to sort, compute resources are not dedicated to processing tasks that do not generate additional value.

II.4.2 Methods of Extracting Attributes

Several variants of the methods exist and deal with the extraction of variables. Among the best known methods are:

Linear Methods

- **Principal Component Analysis (PCA):**
  PCA [41] is an unsupervised linear transformation technique that is primarily used for feature extraction and dimensionality reduction, it aims to find the directions of the maximum variance in large data and projects the data on a new subspace with dimensions equal to or less than the original.

The formula employed to calculate variance (var(x)) and covariance(Cov(x, y)) are expressed as follows:[42]

$$var(x) = \frac{\sum(x_i - \bar{x})^2}{N} \qquad cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

- Var(x) serves as a metric of variability, defines the dataset's degree of dispersion.
- Cov(x, y) captures the covariance between variables x and y
- xi represents the value of x in the ith dimension
- $\bar{x}$ and $\bar{y}$ denote their respective mean values
- The covariance matrix contains:
  1. variance of dimensions as the main diagonal elements
  2. covariance of dimensions as the off-diagonal elements

In the diagram below, note the directions of the maximum variance of the data, this is represented using PCA1 (first maximum variance) and PC2 (second maximum variance).

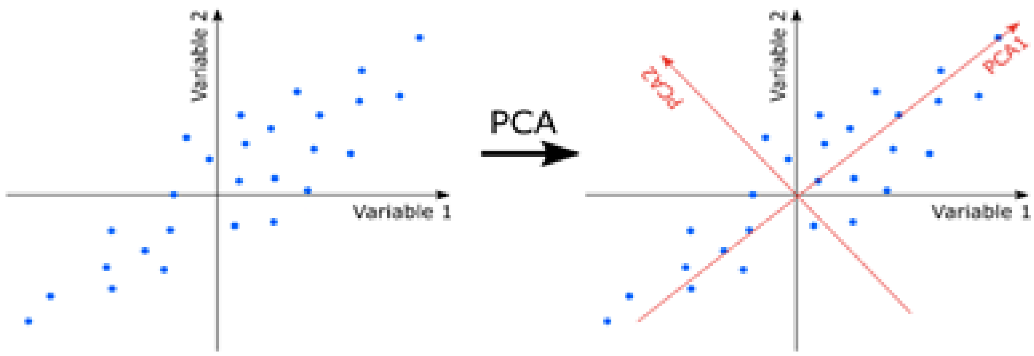


Figure II.7: PCA – Maximum variance directions
[43]

PCA provides good data representation, removes redundancies However, the user may find some difficulties in calculating the covariance and covariance matrix.

- **How PCA Constructs the Principal Components**
  As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. For example, let's assume that the scatter plot of our

data set is as shown below, can we guess the first principal component ? Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).

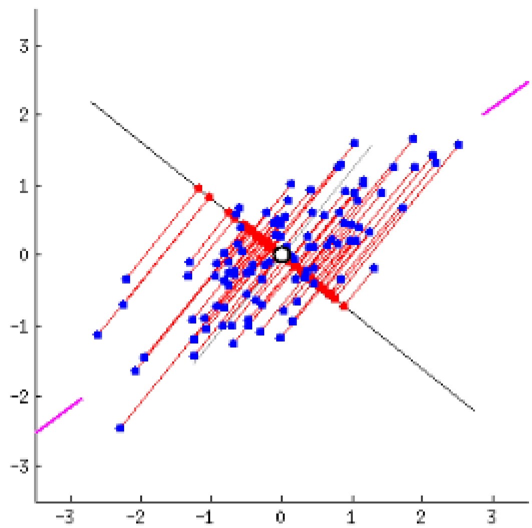


Figure II.8: PCA Construct the Principal Components

The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance. This continues until a total of p principal components have been calculated, equal to the original number of variables.

◇ **Step-by-Step Explanation of PCA**

▷ **Step 1: Standardization** The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis. More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (for example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results, So, transforming the data to comparable scales can prevent this problem. Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{value - mean}{standard\ deviation}$$

Where

$$\text{MEAN} = \frac{Sum\ of\ the\ terms}{Total\ number\ of\ terms}$$

and

$$\text{STANDARD DEVIATION} = \sqrt{\frac{\in(x - mean)^2}{n}}$$

Once the standardization is done, all the variables will be transformed to the same scale.

- **Step 2: Covariance Matrix Computation** The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix. The covariance matrix is a p × p symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables x, y, and z, the covariance matrix is a 3×3 data matrix of this from:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Where

$$\text{Covariance} = \frac{Sum\ (X-(Mean\ of\ X)(Y-(Mean\ of\ Y))}{Number\ of\ data\ points}$$

Since we have standardized the dataset, so the **mean for each feature is 0** and the **standard deviation is 1**.
and :Cov(a,a)=Var(a) | (Cov(a,b)=Cov(b,a))

▷ **Step 3: Compute the eigenvectors and eigenvalues**
Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the principal components of the data. What you first need to know about eigenvectors and eigenvalues is that they always come in pairs, so that every eigenvector has an eigenvalue. Also, their number is equal to the number of dimensions of the data. What you first need to know about eigenvectors and eigenvalues is that they always come in pairs, so that every eigenvector has an eigenvalue. Also, their number is equal to the number of dimensions of the data. By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

Principal Component Analysis Example:
Let's suppose that our data set is 2-dimensional with 2 variables x,y and that the eigenvectors and eigenvalues of the covariance matrix are as follows:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

If we rank the eigenvalues in descending order, we get 1> 2, which means that the eigenvector that corresponds to the first principal component (PC1) is v1 and the one that corresponds to the second principal component (PC2) is v2. After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96 percent and 4 percent of the variance of the data.

▷ **Step 4: Create a Feature Vector**
Example:
Continuing with the example from the previous step, we can either form a feature vector with both of the eigenvectors v1 and v2:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Or discard the eigenvector v2, which is the one of lesser significance, and form a feature vector with v1 only:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

Discarding the eigenvector v2 will reduce dimensionality by 1, and will consequently cause a loss of information in the final data set. But given that v2 was carrying only 4 percent of the information, the loss will be therefore not important and we will still have 96 percent of the information that is carried by v1.

▷ **Step 5: Recast the Data Along the Principal Components Axes**
In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.
$FinalDataSet = \cdot FeatureVector^T * \cdot StandardizedOriginalDataSet^T$

- **Singular Value Decomposition (SVD)** :
  In Machine Learning, one of the most important concepts of linear algebra is singular value decomposition (SVD) [44]. The idea is to break down a matrix into the single product of 3 other matrices. SVD is similar to PCA, but more general.
  PCA assumes that the input matrix is square, while SVD does not have this assumption. The general formula of SVD is:

$$M = U\Sigma V^T$$

  ◇ **M** is the original matrix we want to decompose **M[m*n]** (or a data frame with **m** rows and **n** columns)

  ◇ **U** is **m * m** orthogonal matrix, a left singular values of **M**(columns are left singular vectors). These vectors form an orthogonal basis for the column space of **M**.

  ◇ **Σ** is **m * r** diagonal matrix containing singular values.

  ◇ **V** is a right singular vectors of **M**.These vectors form an orthogonal basis for the row space of **M**.

  ◇ **r** is the rank of the matrix **M**.

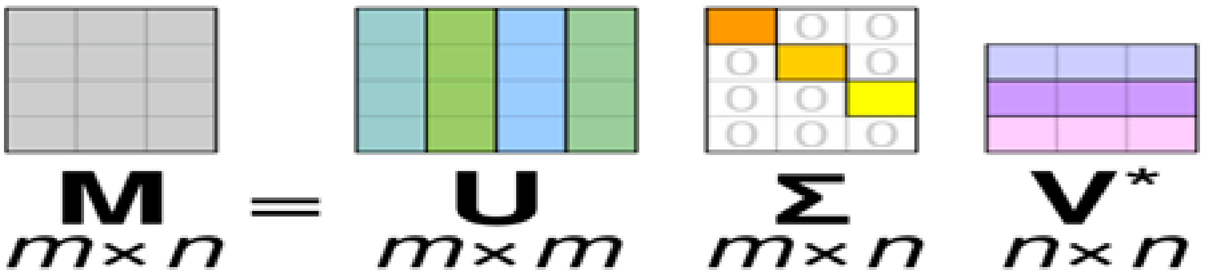


Figure II.9: steps of SVD
[44]

This method is generally used for image compression and data denoising. For dimensionality reduction, **a truncated version of SVD** is often used. Select the top k largest singular values in Σ. These columns can be selected from Σ and the rows selected from V^t. A new matrix B can be reconstructed from the original matrix M using the following formula:

$$B = U * \Sigma$$

$B = V^t * A$, where Σ only contains the top k columns in the original Σ based on singular values and V^t contains the top k rows of the original V^t corresponding to the singular values. SVD allows for dimensionality reduction by retaining only the most significant singular values and vectors but the computing the full SVD for large matrices can be computationally expensive.

- **Linear Discriminant Analysis (LDA):**
  The origin of LDA [45] is different from PCA. PCA is an unsupervised learning method that transforms the original features into a set of new features. LDA is a type of supervised learning technique where the classes of data points are predetermined. LDA computes "linear discriminants" (Where the linear name comes from) determining the directions that serve as axes to maximize separation between multiple classes. The model predicts that all observations in a region belong to the same class of the dependent variable.
  LDA achieves the objective in three main stages:

  1. First, it calculates the separability between the different classes of the dependent variable, called variance between classes, as shown in(1) of **"Figure II.10"**.
  2. Second, it calculates the distance between the mean and the samples of each class, called intra-class variance, as shown in (2).
  3. Then, it constructs the lower dimension space with this criterion: maximize the inter-class variance and minimize the intra-class variance.

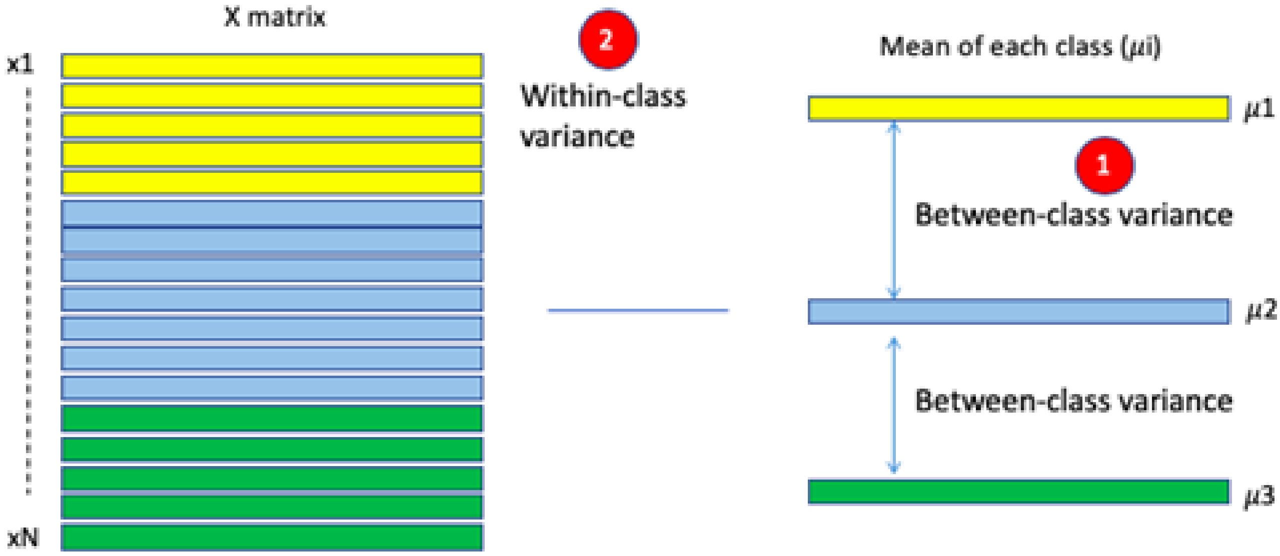


Figure II.10: The LDA feature extraction process
[45]

It can work well even when the number of features is much larger than the number of training samples. However, it assumes that the covariance matrices of the different classes are equal (the features within each class are normally distributed), which may not be true in some datasets.

- **Independent Component Analysis (ICA)**[46]
is a method used for dimensionality reduction, Unlike Principal Component Analysis (PCA), which seeks orthogonal and decorrelated axes that best represent the data, ICA is an unsupervised method that looks for axes that are statistically independent from each other (and therefore decorrelated, but not necessarily orthogonal). While PCA assumes only decorrelation of signals, ICA's concept of independence is stronger. Introduced by Jeanny Herault and Christian Jutten in 1985, ICA was initially used for blind source separation problems but has since found applications in data analysis, compression, Bayesian detection, source localization, and blind identification and deconvolution. While not strictly a dimensionality reduction tool, ICA can effectively reduce dimensions and has been recently used in natural scene classification. In practice, ICA is often combined with PCA, as it requires centered data and PCA preprocessing to obtain a diagonal matrix before transforming data into a space where dimensions are independent.

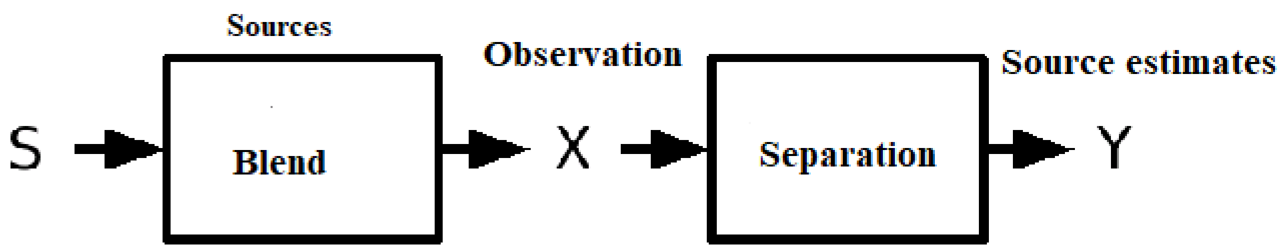


Figure II.11: ICA principle

Nonlinear methods

- **Kernel PCA (KPCA)**
  PCA applies linear transformation, KPCA [47] extends PCA to non-linearity. It first maps the original data to a non-linear (usually larger) feature space, and then applies the PCA to extract the main components of this space (see **Figure II.12**). The graph on the left shows that the blue and red points cannot be separated using a linear transformation. But if all the points are projected on a 3D space, the result becomes linearly separable! We then apply PCA to separate the components.

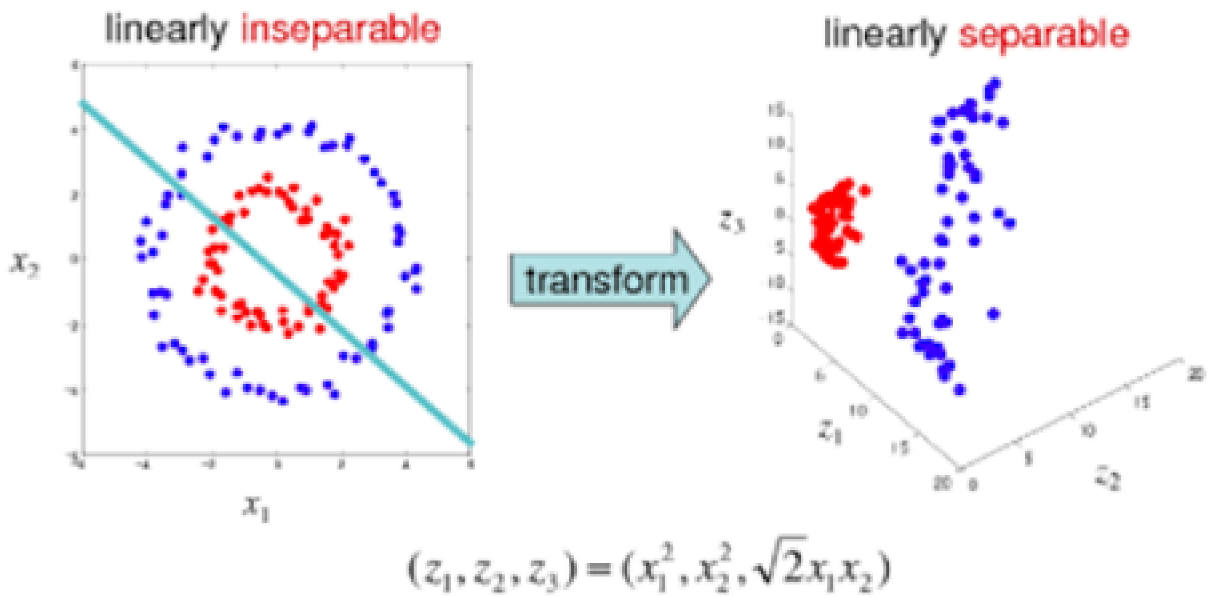


Figure II.12: The process of transforming the original data on a non-linear entity space [45]

KPCA is more complex to implement than classic PCA but it is generally more effective in finding the main directions of the data.

- **Isometric mapping (ISOMAP)**
  A nonlinear dimensionality reduction method used in data analysis and machine learning is called ISOMAP [48], abbreviation for isometric mapping. Isomap was developed to maintain the inherent geometry of high-dimensional data in place of conventional techniques such as principal component analysis (PCA). Isomap creates a low-dimensional representation, usually a two-dimensional or three-dimensional map, focusing on preserving paired distances between data points. This technique works particularly well to extract the underlying structure of large, complex data sets, such as speech recognition, image analysis, and biological systems. Isomap's ability to highlight the fundamental relationships found in the data allows finding models and ideas in a variety of scientific and technical fields.

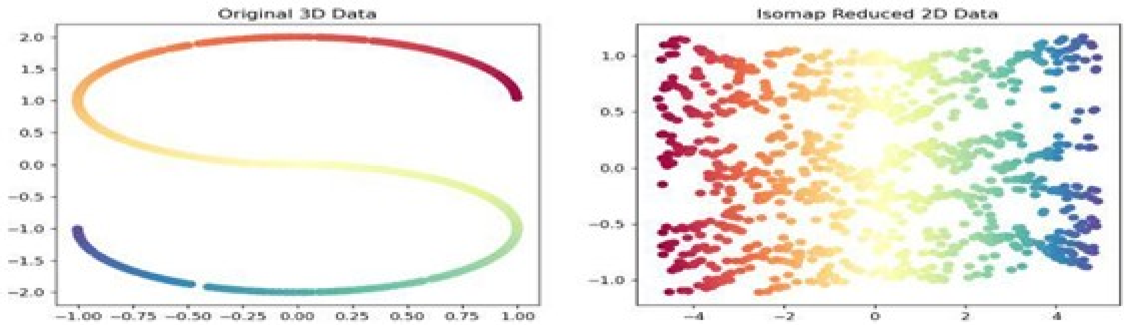


Figure II.13: before and after application Isomap

- **Integration of stochastic neighbors distributed by t (t-SNE):**
  t-Distributed Stochastic Neighbor Embedding or t-SNE [49]is a dimensionality reduction technique well suited for data visualization. Contrary to PCA which simply maximizes the variance, t-SNE minimizes the divergence between two distributions. Essentially, it recreates the distribution of a high-dimensional space in a low-dimensional space rather than maximizing variance or even using a kernel trick. We can get a high-level understanding of t-SNE in three simple steps:
  - It first creates a probability distribution for the high-dimensional samples.
  - Then, it defines a similar distribution for the points in the low-dimensional embedding.
  - Finally, it tries to minimize the KL-divergence between the two distributions.

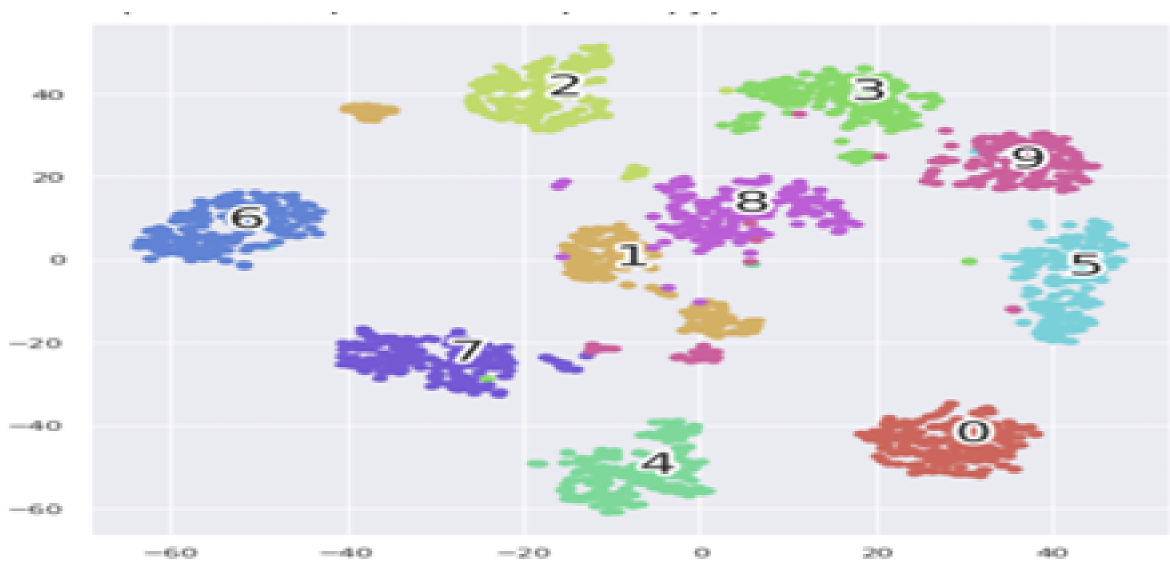


Figure II.14: Dimensionality reduction technique: t-SNE [49]

II.4.3 Comparison between Features Extraction methods

PARAMETERS	PCA	LDA	SVD	ICA	KPCA
Data preprocessing	Not required	Not required	Required	Not required	For large data set Required
Dataset type	Eigen values	-	Multivariate data, gene expression data	Multivariate Data.	Eigen values
Fault Tolerance	Less Sensitive due to Linear Nature	Less sensitive	Less sensitive	Sensitive to Fault	More sensitive to fault compare to PCA because of nonlinear Behavior
Large data set handling ability	Good	Good	Not good	Good	Moderate
Multidimensional Data set ability	Good	Good	Not good	Good	Very good
Training	Required	Not required	Not required	Not required	Not required
Training time	High	Less than PCA	Moderate	Slightly moderate compare to another model	Very High

Figure II.15: Comparison of different Dimensionality Reduction Methods [50]

II.5 difference between Feature Selection and Feature Extraction

Feature Selection	Feature Extraction
Selects a subset of relevant features from the original set of features.	Extracts a new set of features that are more informative and compact.
Reduces the dimensionality of the feature space and simplifies the model.	Captures the essential information from the original features and represents it in a lower-dimensional feature space.
Can be categorized into filter, wrapper, and embedded methods.	Can be categorized into linear and nonlinear methods.
Requires domain knowledge and feature engineering.	Can be applied to raw data without feature engineering.
Can improve the model's interpretability and reduce overfitting.	Can improve the model performance and handle nonlinear relationships.
May lose some information and introduce bias if the wrong features are selected.	May introduce some noise and redundancy if the extracted features are not informative.

Figure II.16: comparison between Feature Selection and Feature Extraction [51]

II.6 Conclusion

In this chapter, we mentioned Dimensionality Reduction and its role in dealing with big data (which contains many numbers of attributes). DR consists of two main methods, 'Features selection and Features extraction'. Both have the main task, which leads to reduced dataset dimensions. Each technique has different algorithms that are Probably applied to different datasets. In the next chapter, we will study feature extraction closely while testing some methods, including inferring their importance in improving classifier performance

Chapter III

Contribution and Implementation

III.1 Introduction

In this chapter, we discuss the crucial process of feature extraction in machine learning. Feature extraction is a fundamental step that aims to identify and meaningfully represent the essential information contained in a data set. This simplifies the complexity of the data while preserving the most relevant aspects for the construction of efficient models. When working with large or complex data sets, the presence of many features can be challenging. Feature extraction aims to select the most informative aspects while eliminating noise or redundancies, making the task of machine learning algorithms easier. In the case of Malaware detection; the datasets contain observations with missing values, unequal distribution between classes, and specially a **multitude of variables**. The extraction of the characteristics then becomes crucial to prepare the data for the construction and interpretation of the models. In this chapter, we will explore the different techniques of feature extraction, focusing on their impact on model performance. We will evaluate these techniques using several performance measures such as accuracy, recall, and we will also consider the time required for these operations.

III.2 Datasets

Is a valuable resource for researchers interested in studying malware detection on Android devices. It provides comprehensive information about the different types of permissions associated with Android apps and can be used to train machine learning algorithms to detect malicious apps. Among the most important variables in the dataset are:

- App Permissions: represents the different permissions requested by each Android app.
- Application type: indicates whether an application is goodware or malware.
- Package name: represents the unique name assigned to each Android app package.
- File size: the size of the application installation file.
- Minimum version of the SDK: minimum version of Android required for the application to work.
- Target SDK version: the version of Android for which the application was developed.

III.2.1 The datasets used

1. **DREBIN**
   Dataset consisting of feature vectors of 215 attributes extracted from 15,036 applications (5,560 malware apps from Drebin project and 9,476 benign apps)[52], The dataset has been used to develop and evaluate multilevel classifier fusion approach for Android malware detection.

class distribution	number of observations	number of variables	missing values	Missing cells (%)	Total size in memory
malware: 5560 goodware: 9476	15036	216	0	0	24.78 MB

Figure III.1: DREBIN-215 dataset details

2. **TUANDROMD**
   Tundromd dataset contains 4465 instances and 241 attributes. The target attribute for classification is a category (malware vs goodware), (N.B: This is the preprocessed version of Tundromd)[53]

class distribution	number of observations	number of variables	missing values	Missing cells (%)	Total size in memory
malware: 3565 goodware: 899	4465	242	0	0	24.78 MB

Figure III.2: TUANDROMD dataset details

3. **Malgenome**
   Dataset consisting of feature vectors of 215 attributes extracted from 3799 applications (1260 malware apps from Android malgenome project and 2539 benign apps). The dataset has been used to develop and evaluate multilevel classifier fusion approach for Android malware detection.[54]

CLASS DISTRIBUTION	NUMBER OF OBSERVATIONS	NUMBER OF VARIABLES	MISSING VALUES	MISSING CELLS (%)	TOTAL SIZE IN MEMORY
MALWARE:2539 GOODWARE: 1260	3799	216	0	0	6.26 MB

Figure III.3: Malgenome dataset details

III.3 The Implementation Tools

III.3.1 JupyterLab

JupyterLab is the latest web-based interactive development environment for code, data notebooks. JupyterLab is a simple interface which allows users to configure and arrange workflows in machine learning, scientific computing, computational journalism, and data science.[55]

III.3.2 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics, Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance, Python supports modules and packages, which encourages program modularity and code reuse, the Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed [56].

III.3.3 Dataset management (libraries)

The main libraries used in the code are:

- ◇ **Pandas**
  Pandas is a popular Python library for data manipulation and analysis. It provides data structures and functions for efficiently handling and analyzing structured data, primarily in the form of tables or DataFrames. Pandas is widely used in data science, data analysis, and data preprocessing tasks[57]. we used this library to store our dataset in memory.

- ◇ **NumPy**
  NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices, NumPy was created in 2005 by Travis Oliphant, it is an open-source project and you can use it freely, NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. NumPy stands for Numerical Python.[58]

- ◇ **Matplotlib**
  Matplotlib is a comprehensive library for creating static, animated, and

interactive visualizations in Python, Matplotlib makes easy things easy and hard things possible.[59]

◇ **Scikit-Learn**

Scikit-Learn is a Python library specialized in Data Science work. It is an easily accessible, powerful library that fits naturally into the broader ecosystem of Python-based data science tools.
Scikit-learn provides a variety of supervised and unsupervised algorithms: [60]

- ▷ **SVC ():** A support vector classifier (SVC) from the scikit-learn library (sklearn) that performs classifications using support vectors in high-dimensional space.
- ▷ neighbors.KNeighborsClassifier(): A supervised learning algorithm from the scikit-learn library (sklearn) that performs classifications based on the k closest examples in feature space.
- ▷ **tree.DecisionTreeClassifier():** A supervised learning algorithm from the scikit-learn library (sklearn) that builds a decision tree from training data to perform classifications
- ▷ **RandomForestClassifier():** A supervised learning algorithm from the scikit-learn library (sklearn) that builds an ensemble of multiple decision trees and then uses majority voting to make classifications.
- ▷ **traintestsplit** splitting the dataset into training and testing): A function in the scikit-learn library (sklearn) that allows you to split a dataset into two distinct parts: a training set used to fit the model and a training set used to tune the model and a test used to evaluate the performance of the model.

III.4 Performance Evaluation

Regardless of the type of learning used, after the learning phase, a template will be created. It is necessary to verify the proper functioning and generalization of this model. Evaluating the prediction of a model with the same data that was used for learning is not useful. To properly evaluate a model, it must be tested on data that was not part of the learning data. Prediction results should be compared to values of known results.

III.4.1 Validation Methods

To validate learning models correctly, two validation methods are used: sampling and cross validation.[22]

- **Sampling:**
  Sampling consists of dividing the collected set of data into two parts: one for learning and the other for testing. Different sampling techniques are used depending on the nature and size of the data set: random, rejection and preferential, **Figure III.4** shows a simple example of a sample where the data set of 16 individuals is divided into two equal parts; one for learning (8 individuals) and one for testing (8 individuals).[61]

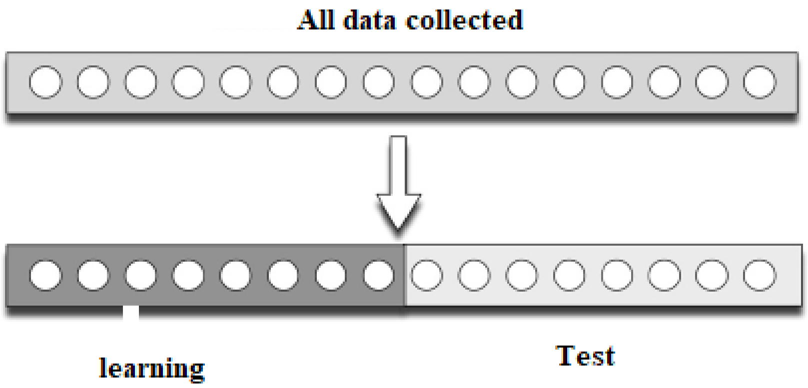


Figure III.4: Example on sampling

- **Cross-Validation**
  Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance. Cross validation is an important step in the machine learning process and helps to ensure that the model selected for deployment is robust and generalizes well to new data.[61]

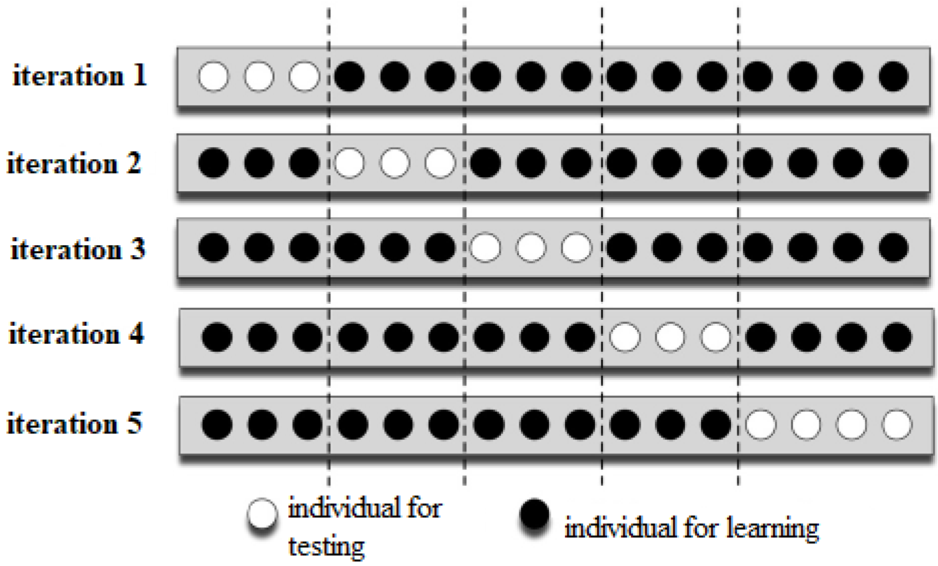


Figure III.5: Example on cross validation

III.4.2 Performance Measures

Confusion matrix

A confusion matrix, also called an error matrix, is an N x N matrix used to evaluate the performance of a classification model, where N is the number of target classes, The matrix compares the actual target values with those predicted by the machine learning model, This gives us an overall view of the performance of our classification model and the types of errors it makes, It has 4 essential values [61]

		Prediction	
		C1	C2
Actual values	C1	True positive (VP)	False negative (FN)
	C2	C2 False positive (FP)	True negative (VN)

Figure III.6: Confusion matrix

According to the confusion matrix: the true positive (VP) indicates the number of individuals in the validation set who are correctly classified in C1, unlike the false negative (FN) which indicates the number of individuals in C1 who are misclassified in C2m A true negative (VN) shows the number of individuals who are correctly classified in C2 while the false positive (FP) indicates the number of individuals in C2 who are misclassified in C1, The following is a list of measures generally adopted for model comparison and validation:

- **Accuracy**
  The accuracy parameter determines the correct prediction rate among all positive and negative classes.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **precision**
  This is the proportion of the individuals of ci that were actually correctly identified by the model.

$$\text{Precision:} = \frac{TP}{TP+FP}$$

- **Recall:**
  This is the proportion of class Ci individuals that were actually identified by the mode.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **F1-score**
  This is the harmonic mean between precision and recall.

$$\text{F1} = \frac{2 » Precision » Recall}{Precision+Recall}$$

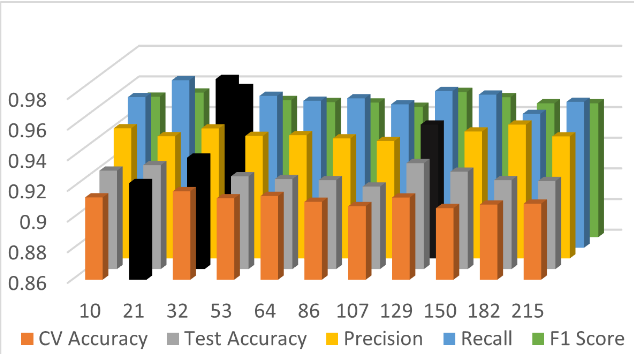
III.5 Extraction techniques

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Independent Component Analysis(ICA)
- Kernel PCA (KPCA)

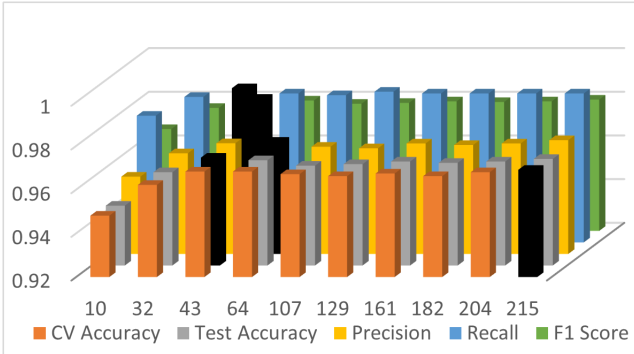
III.6 Experimentations

III.6.1 DATASET DREBIN

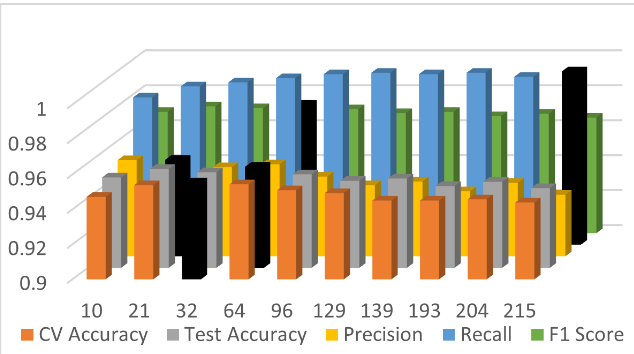
Method 01: Principal Component Analysis (PCA)



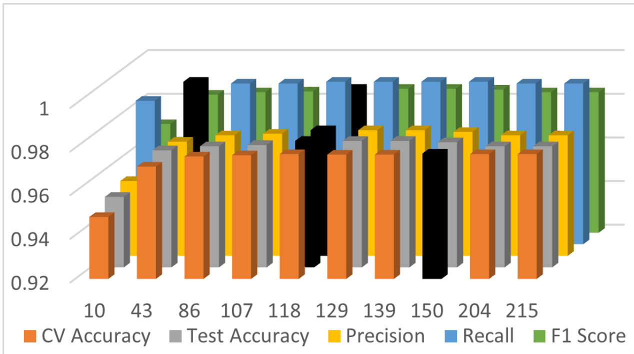
G1: Decision Tree Classifier



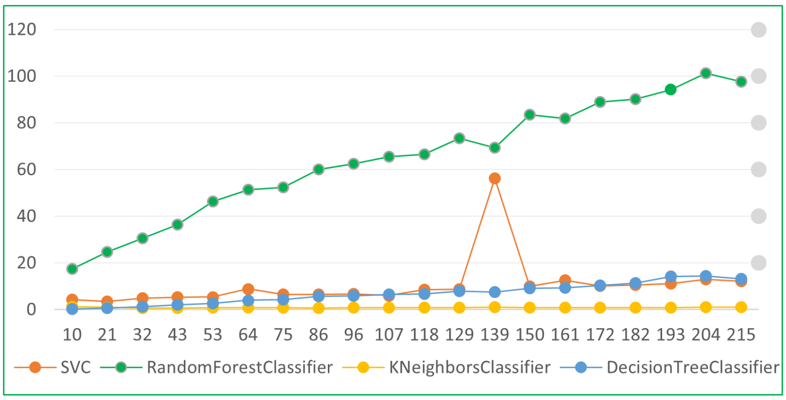
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



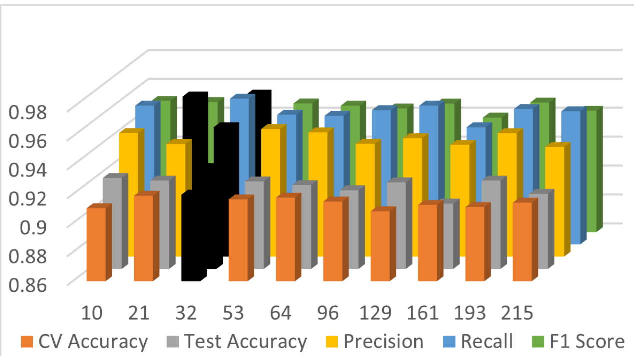
The result of the PCA method for Drebin dataset

Best results

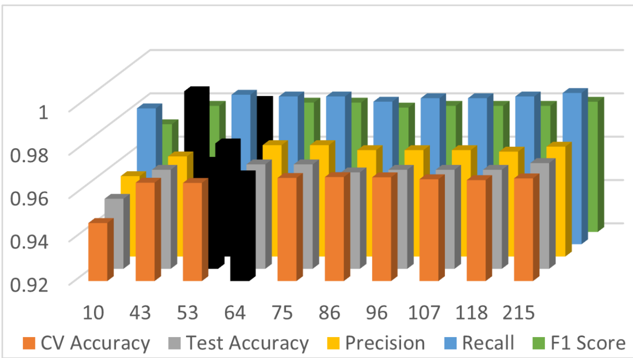
	DREBIN PCA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	15 %	0,9573	20%	0,9805	30%	0,9735	55%	0,986
Recall	15%	0 ,9701	20%	0,9906	30%	0.9953	55%	0.9945
Precision	15%	0.9448	20%	0.9707	30%	0.9526	55%	0,9776
Accuracy	15%	0,9327	20%	0,9694	30%	0,9578	55%	0,9787

Discussion:
Application of PCA on Derbin dataset has improved the performance of classifiers with reduction in the number of attributes (from 15% to 55%) we obtained the best results:
DecisionTreeClassifier:
With 19 attrubites (15%) among 215, we obtained an F1-Score 95.73% and Accuracy 93.27%
KNeighborsClassifier
With 25 attrubites (20%) among 215, we obtained an F1-Score 98.05% and Accuracy 96.94%
RandomForestClassifier
With 38 attrubites (30%) among 215, we obtained an F1-Score 97.35% and Accuracy 97.78%
SVC
With 70 attrubites (55%) among 215, we obtained an F1-Score 98.6% and Accuracy 97.87%

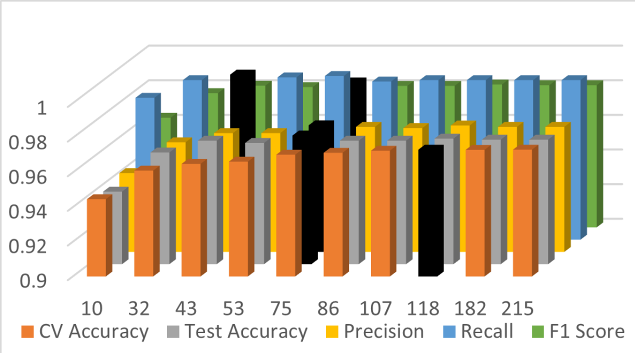
Method 02: Singular Value Decomposition (SVD)



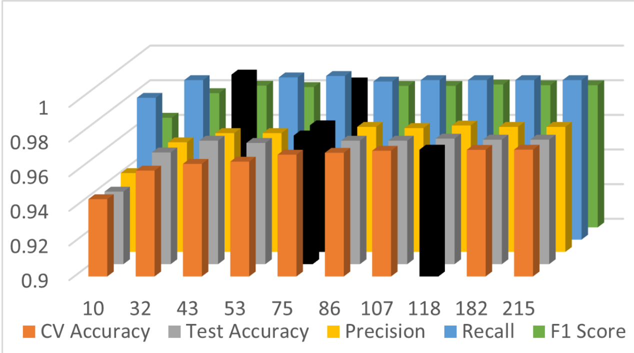
G1: Decision Tree Classifier



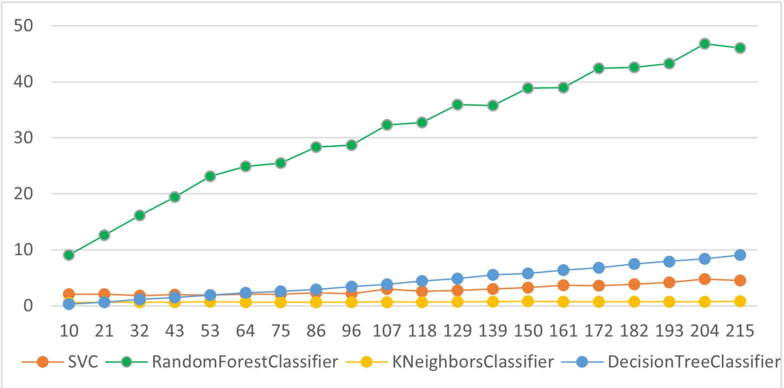
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the SVD method for Drebin dataset

Best results

	DREBIN SVD							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	15%	0,955	25%	0,9805	15%	0,9733	35%	0,9837
Recall	15%	0.9607	25%	0.989	15%	0.9913	35%	0.9945
Precision	15%	0,9495	25%	0,9722	15%	0.956	35%	0,9731
Accuracy	15%	0,9297	25%	0,9694	15%	0,9578	35%	0,9743

Discussion:

The best results with SVD are observed between 15% and 35% of attributs:

DecisionTreeClassifier:

With 19 attrubites 15% among 215, we obtained an F1-Score 95.5% and Accuracy 92.97%

KNeighborsClassifier

With 25 attrubites 20% among 215, we obtained an F1-Score 98.05% and Accuracy 96.94%

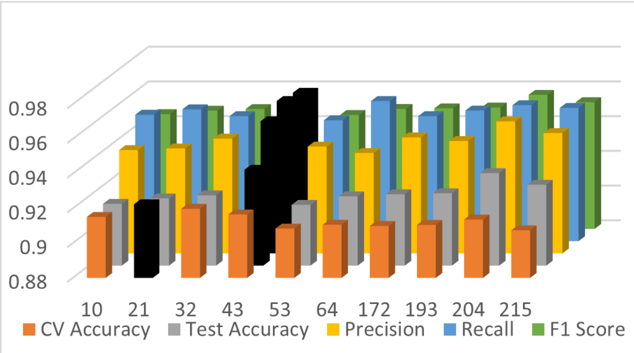
RandomForestClassifier

WIth 38 attrubites 30% among 215, we obtained an F1-Score 97.33% and Accuracy 95.78%

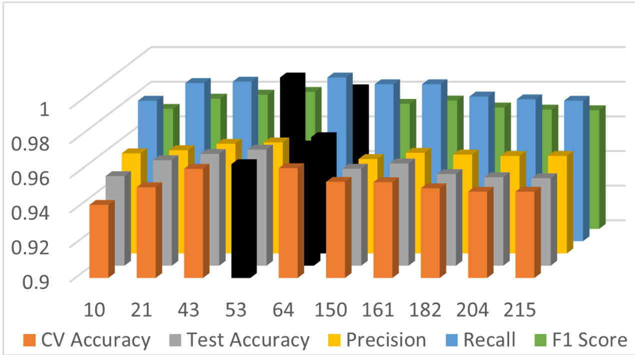
SVC

With 70 attrubites 55% among 215, we obtained an F1-Score 98.37% and Accuracy 97.43%

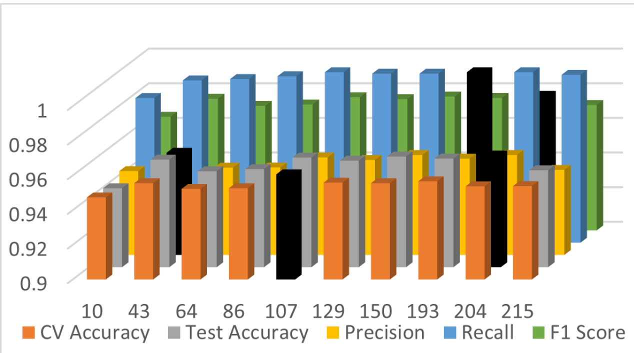
Method 03: Independent Component Analysis(ICA)



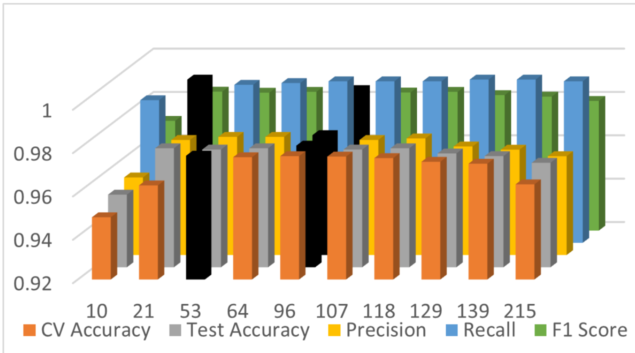
G1: Decision Tree Classifier



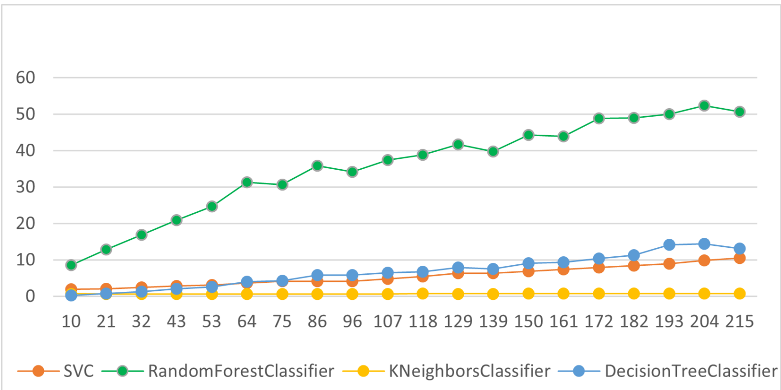
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the ICA method for Drebin dataset

Best results

	DREBIN ICA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	20%	0,9584	30%	0,9806	95%	0,9777	45%	0,9848
Recall	20%	0,9607	30%	0.9945	95%	0.9984	45%	0.9945
Precision	20%	0,9561	30%	0,9671	95%	0.9577	45%	0,9753
Accuracy	20%	0,9352	30%	0,9694	95%	0,9645	45%	0,9761

Discussion:

ICA has also improved the performance of the different classifier. We have obtained :

DecisionTreeClassifier:

With 25 attrubites 20% among 215, we obtained an F1-Score 95.84% and Accuracy 93.52%

KNeighborsClassifier

With 38 attrubites 30% among 215, we obtained an F1-Score 98.06% and Accuracy 96.94%

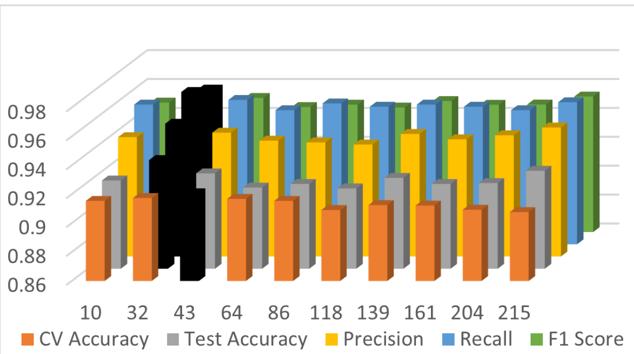
RandomForestClassifier

With 122 attrubites 95% among 215, we obtained an F1-Score 97.77% and Accuracy 96.45%

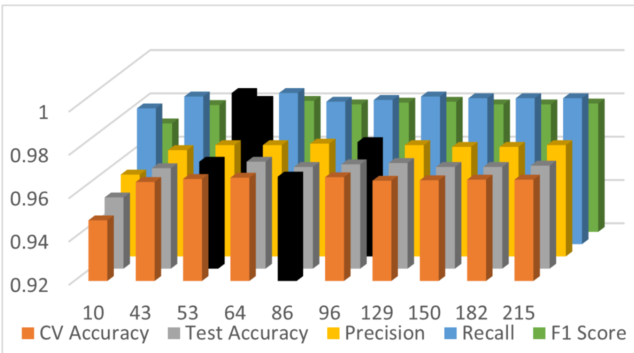
SVC

With 58 attrubites 45% among 215, we obtained an F1-Score 98.48% and Accuracy 97.61%

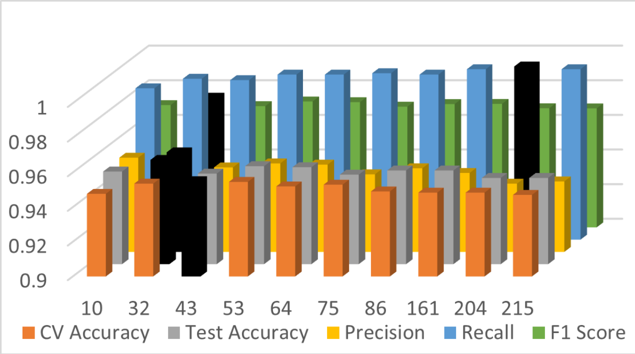
Method 04: Kernel PCA (KPCA)



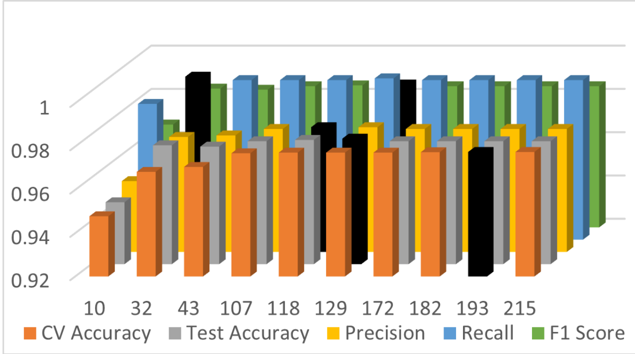
G1: Decision Tree Classifier



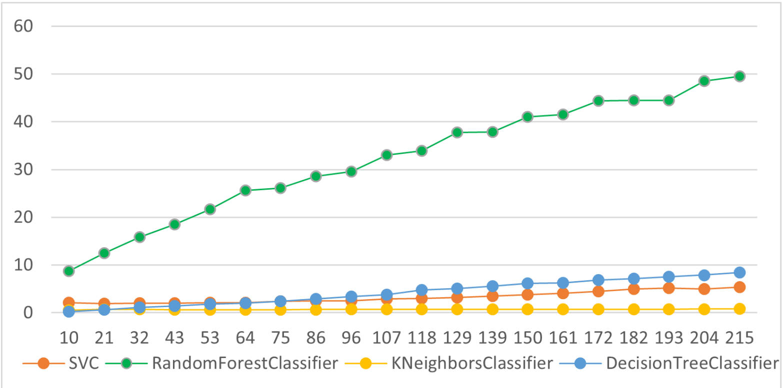
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the KPCA method for Drebin dataset

Best results

	DREBIN KPCA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	15%	0,9586	25%	0,9805	15%	0,9749	60%	0,986
Recall	15%	0,9654	25%	0,9898	15%	0.9929	60%	0.9945
Precision	15%	0,9519	25%	0.9714	15%	0,9575	60%	0.9776
Accuracy	15%	0,9352	25%	0,9694	15%	0,9602	60%	0,978

Discussion:

With KPCA we have also noticed an improvement in results:

DecisionTreeClassifier:

With 19 attrubites 15% among 215, we obtained an F1-Score 95.86% and Accuracy 93.52%

KNeighborsClassifier

With 32 attrubites 25% among 215, we obtained an F1-Score 98.05% and Accuracy 96.94%

RandomForestClassifier

With 19 attrubites 15% among 215, we obtained an F1-Score 97.49% and Accuracy 96.02%

SVC

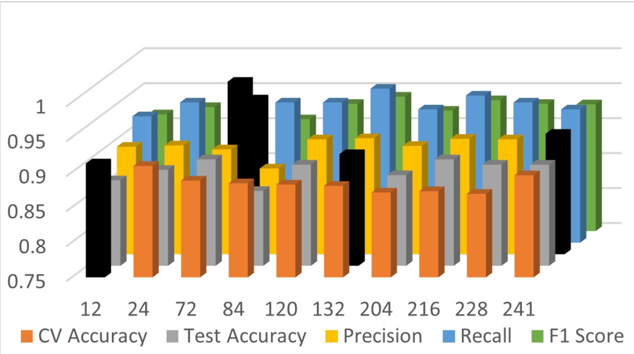
With 77 attrubites 60% among 215, we obtained an F1-Score 98.6% and Accuracy 97.8%

Synthesis:
the best method according to accuracy, F1 score, and low percentage is Kernel PCA (KPCA)
F1 Score: 98.6%
Accuracy: 97.8%

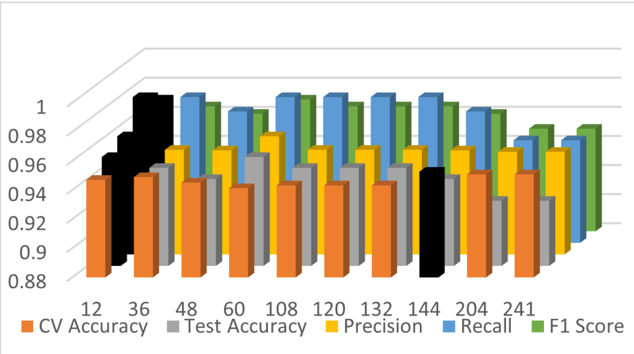
Second-Best Method is Independent Component Analysis (ICA) with:
Accuracy: 97.61%
F1 Score: 98.48%

III.6.2 DATASET TUANDROMD

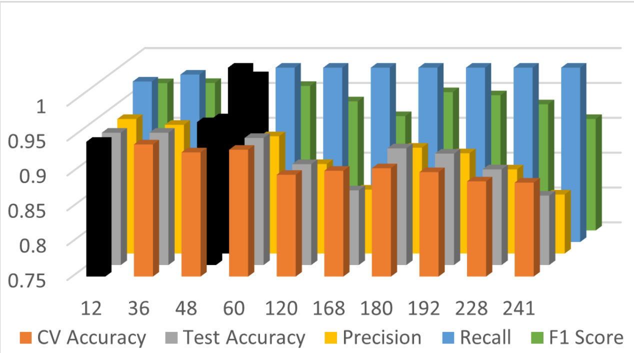
Method 01: Principal Component Analysis (PCA)



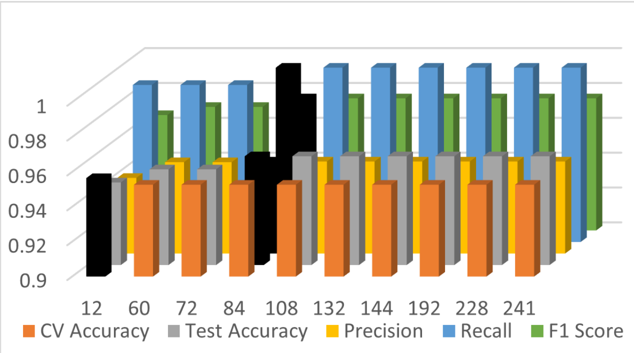
G1: Decision Tree Classifier



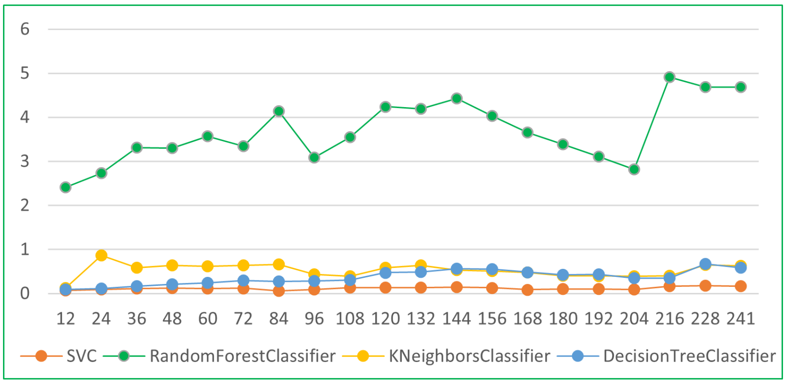
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the PCA method for Tuandromd dataset

Best results

	TUANDROMD PCA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	55%	0,9423	5%	0,9706	20%	0,9712	35%	0,9758
Recall	55%	0,9703	5%	0,9802	20%	1	35%	1
Precision	55%	0.9159	5%	0,9612	20%	0,9439	35%	0,9528
Accuracy	55%	0,9098	5%	0,9549	20%	0,9549	35%	0,9624

Discussion:

Application of PCA on Tuandromd dataset has improved the performance of classifiers with reduction in the number of attributes (from 5% to 55%) we obtained the best results:

DecisionTreeClassifier:

With 132 attrubites 55% among 215, we obtained an F1-Score 94.23% and Accuracy 90.98%

KNeighborsClassifier

With 12 attrubites 5% among 215, we obtained an F1-Score 97.06% and Accuracy 95.49%

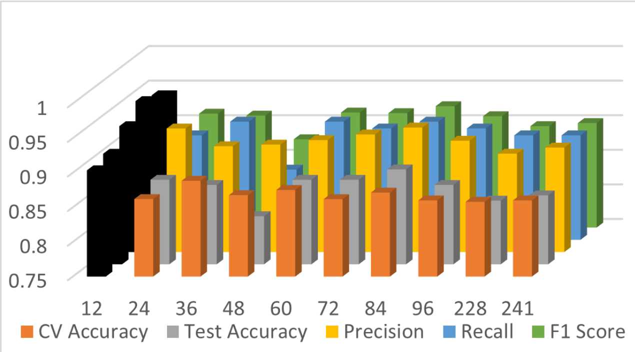
RandomForestClassifier

With 48 attrubites 20% among 215, we obtained an F1-Score 97.12% and Accuracy 95.49%

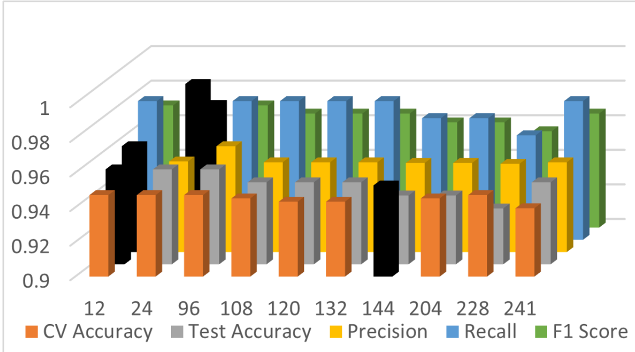
SVC

With 84 attrubites 35% among 215, we obtained an F1-Score 97.58% and Accuracy 96.8724%

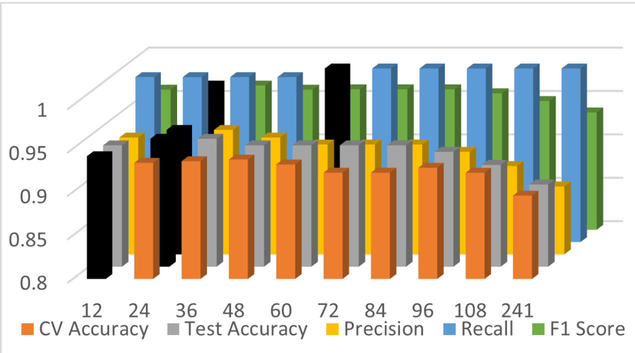
Method 02: Singular Value Decomposition (SVD)



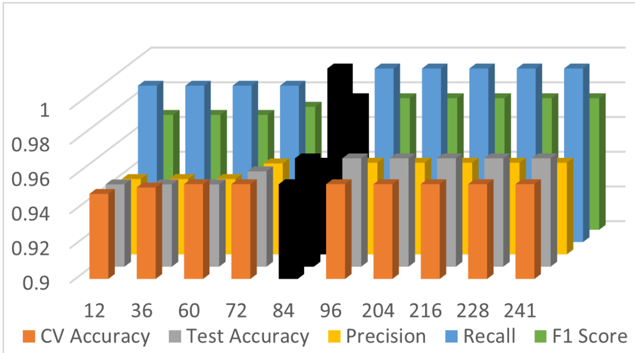
G1: Decision Tree Classifier



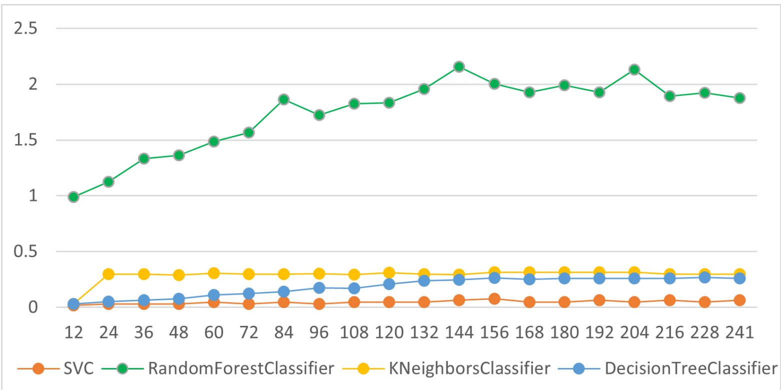
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



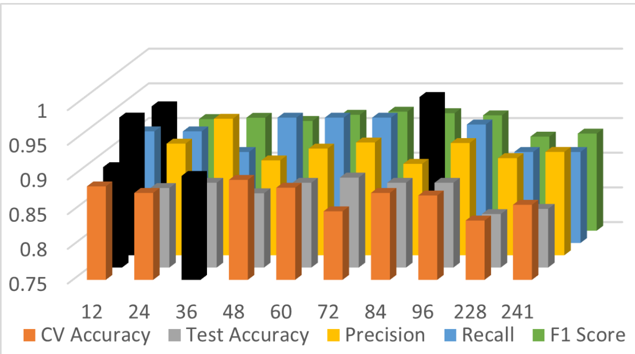
The result of the SVD method for Tuandromd dataset

Best results

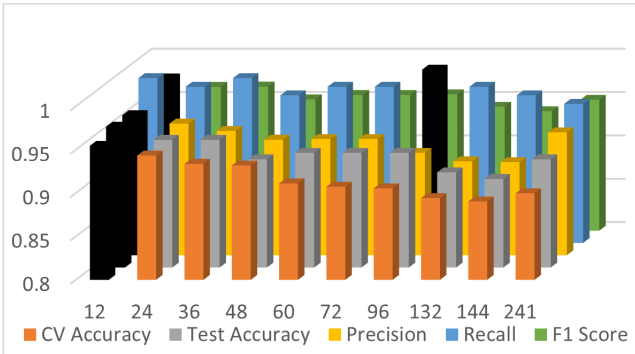
	TUANDROMD SVD							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	5%	0,9412	5%	0,9709	10%	0,9662	35%	0,9758
Recall	5%	0,9505	5%	0,9901	10%	0.9901	35%	1
Precision	5%	0,932	5%	0,9612	10%	0,9434	35%	0,9528
Accuracy	5%	0,9098	5%	0,9549	10%	0,9474	35%	0,9624

Discussion:
The best results with SVD are observed between 15% and 35% of attributs:
DecisionTreeClassifier:
With 12 attrubites 5% among 215, we obtained an F1-Score 94.12% and Accuracy 90.98%
KNeighborsClassifier
With 12 attrubites 5% among 215, we obtained an F1-Score 97.09% and Accuracy 95.49%
RandomForestClassifier
With 24 attrubites 10% among 215, we obtained an F1-Score 96.62% and Accuracy 94.74%
SVC
With 84 attrubites 35% among 215, we obtained an F1-Score 97.58% and Accuracy 96.24%

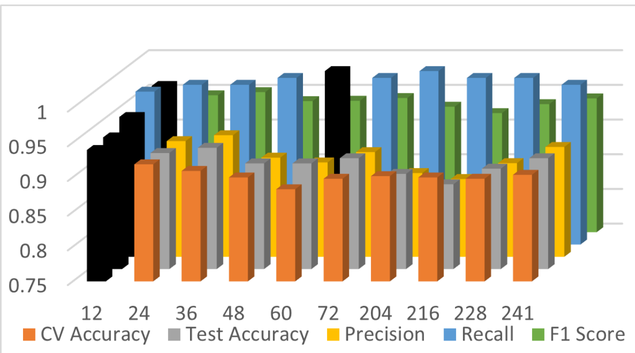
Method 03: Independent Component Analysis(ICA)



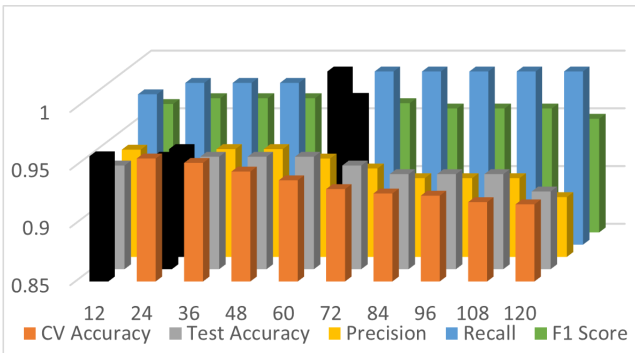
G1: Decision Tree Classifier



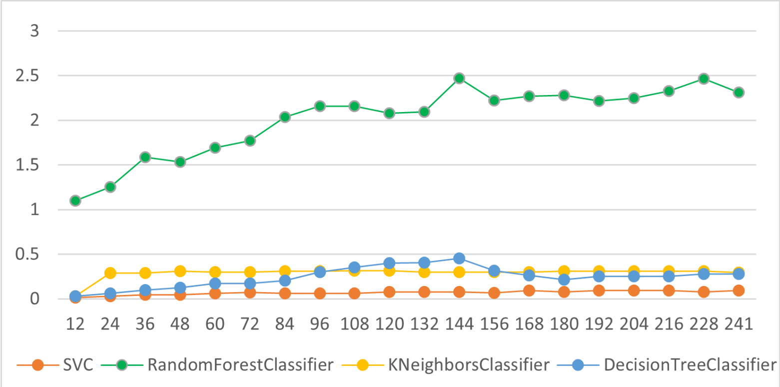
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



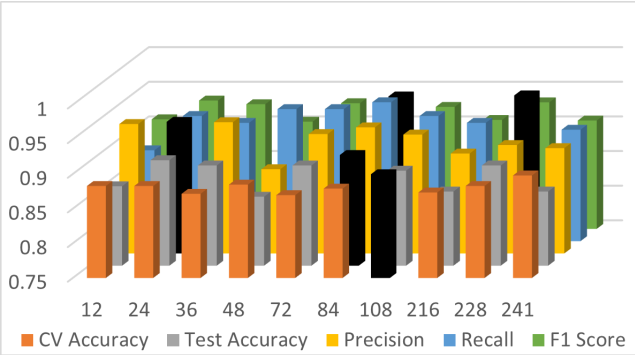
The result of the ICA method for Tuandromd dataset

Best results

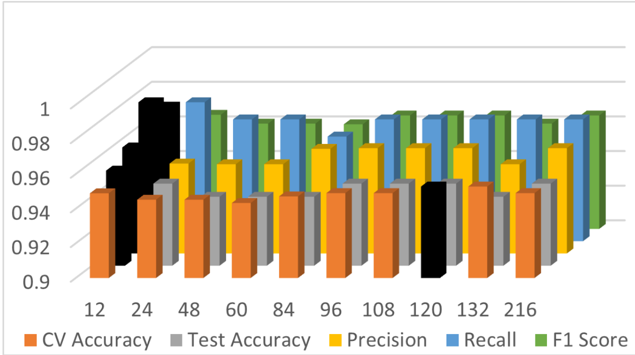
	TUANDROMD ICA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	5%	0,9293	5%	0,9756	5%	0,9608	10%	0,9662
Recall	5%	0.9109	5%	0.9901	5%	0.9703	10%	0.9901
Precision	5%	0,9485	5%	0,9615	5%	0,9515	10%	0,9434
Accuracy	5%	0,8947	5%	0,9624	5%	0,9398	10%	0,9474

Discussion:
ICA has also improved the performance of the different classifier. We have obtained:
DecisionTreeClassifier:
With 12 attrubites 5% among 215, we obtained an F1-Score 92.93% and Accuracy 89.47%
KNeighborsClassifier
With 12 attrubites 5% among 215, we obtained an F1-Score 97.56% and Accuracy 96.24%
RandomForestClassifier
With 12 attrubites 10% among 215, we obtained an F1-Score 96.08% and Accuracy 93.98%
SVC
With 24 attrubites 10% among 215, we obtained an F1-Score 96.6% and Accuracy 94.74%

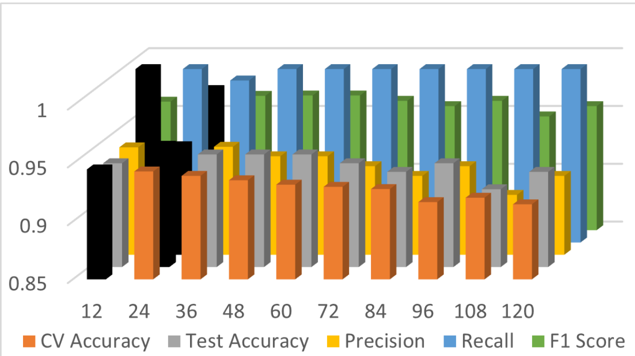
Method 04: Kernel PCA (KPCA)



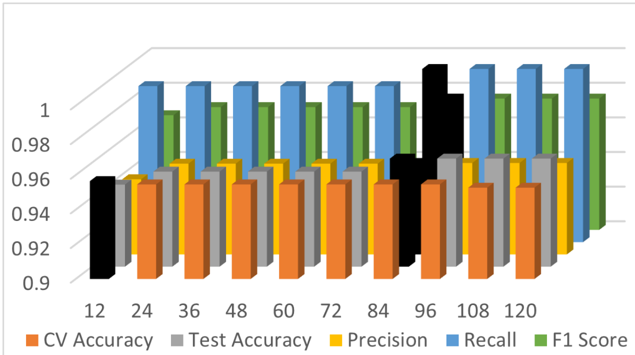
G1: Decision Tree Classifier



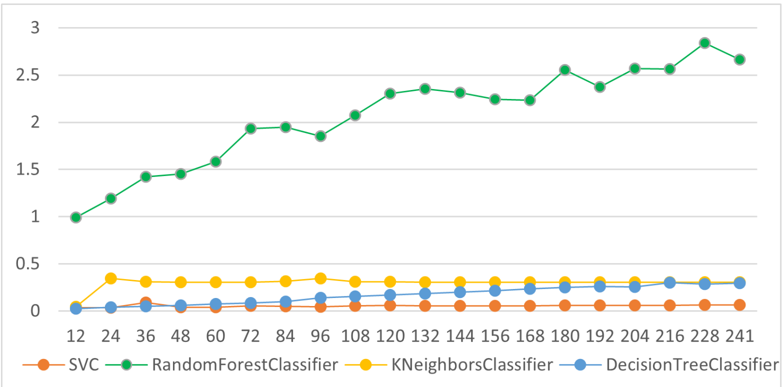
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the KPCA method for Tuandromd dataset

Best results

	TUANDROMD KPCA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	35%	0,9412	5%	0,9706	10%	0,9712	35%	0,9758
Recall	35%	0.9505	5%	0,9802	10%	1	35%	1
Precision	35%	0.932	5%	0,9612	10%	0,9439	35%	0,9528
Accuracy	35%	0,9098	5%	0,9549	10%	0,9549	35%	0,9624

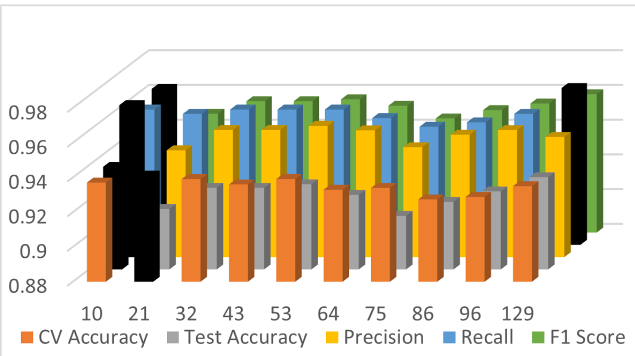
Discussion:
With KPCA we have also noticed an improvement in results:
DecisionTreeClassifier:
With 84 attrubites 35% among 215, we obtained an F1-Score 94.12% and Accuracy 90.98%
KNeighborsClassifier
With 12 attrubites 5% among 215, we obtained an F1-Score 97.06% and Accuracy 95.49%
RandomForestClassifier
With 24 attrubites 10% among 215, we obtained an F1-Score 97.12% and Accuracy 95.59%
SVC
With 84 attrubites 35% among 215, we obtained an F1-Score 97.58% and Accuracy 96.24%

Synthesis:
the best method according to accuracy, F1 score, and low percentage is Kernel PCA (KPCA)
Accuracy: 96.24% **F1 Score:** 97.58%

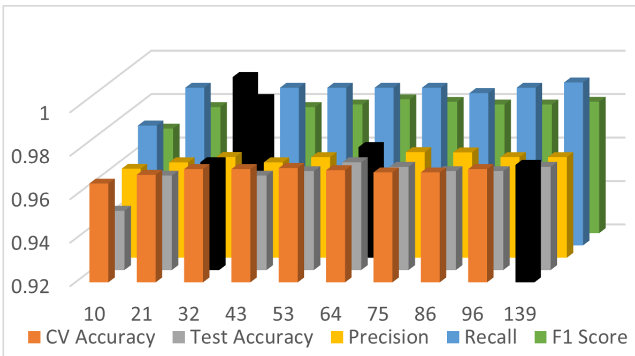
Second-Best Method is Independent Component Analysis (ICA) with:
F1 Score: 97.56%
Accuracy: 96.24%

III.6.3 DATASET MALGENOME

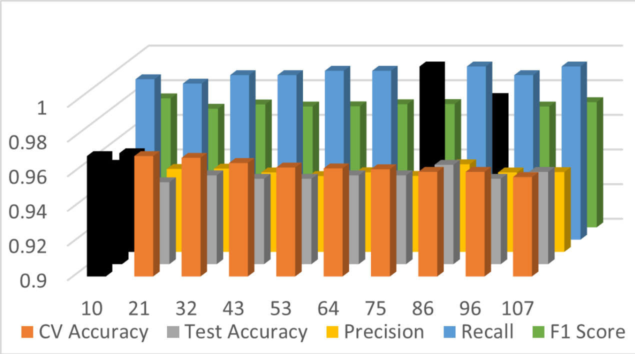
Method 01: Principal Component Analysis (PCA)



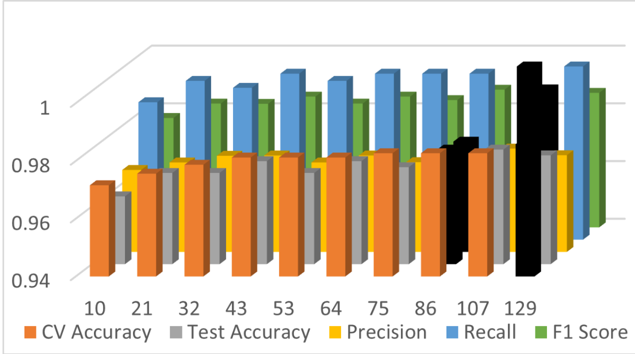
G1: Decision Tree Classifier



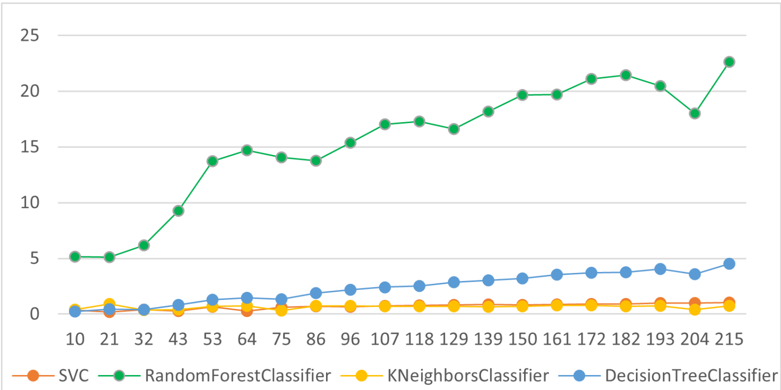
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the PCA method for Malgenome dataset

Best results

	MALGENOME PCA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	5%	0,9627	15%	0,9817	5%	0.9745	40%	0,9877
Recall	5%	0.9576	15%	0,9975	5%	0.9926	40%	0,9975
Precision	5%	0,9675	15%	0,9709	5%	0,957	40%	0,9782
Accuracy	5%	0,9391	15%	0.9696	5%	0,9574	40%	0,9797

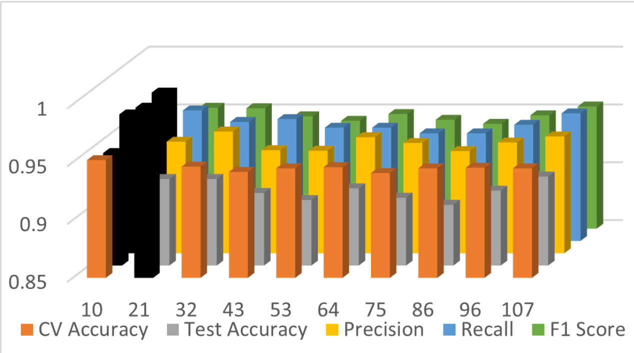
Discussion:

Application of PCA on Malgenome dataset has improved the performance of classifiers with reduction in the number of attributes (from 5% to 40%) we obtained the best results: **DecisionTreeClassifier:**
With 10 attrubites 5% among 215, we obtained an F1-Score 96.27% and Accuracy 93.91%
KNeighborsClassifier
With 32 attrubites 15% among 215, we obtained an F1-Score 98.17% and Accuracy 96.96%
RandomForestClassifier
With 10 attrubites 5% among 215, we obtained an F1-Score 97.45% and Accuracy 95.74%

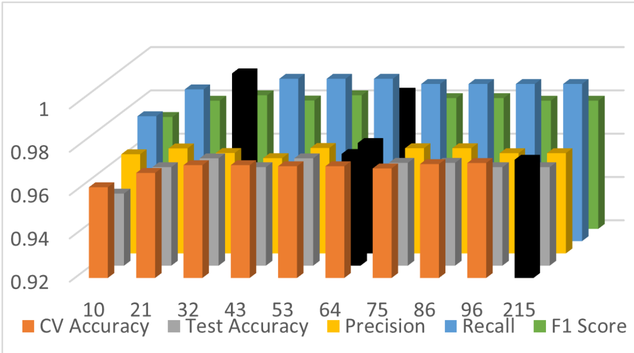
SVC

With 86 attrubites 40% among 215, we obtained an F1-Score 98.77% and Accuracy 97.97%

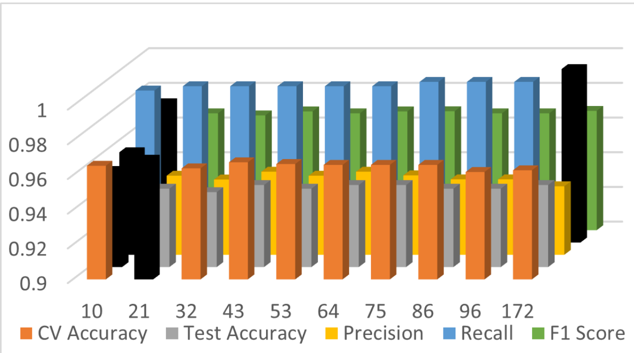
Method 02: Singular Value Decomposition (SVD)



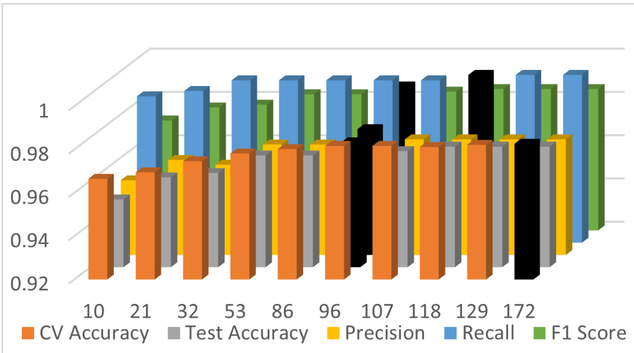
G1: Decision Tree Classifier



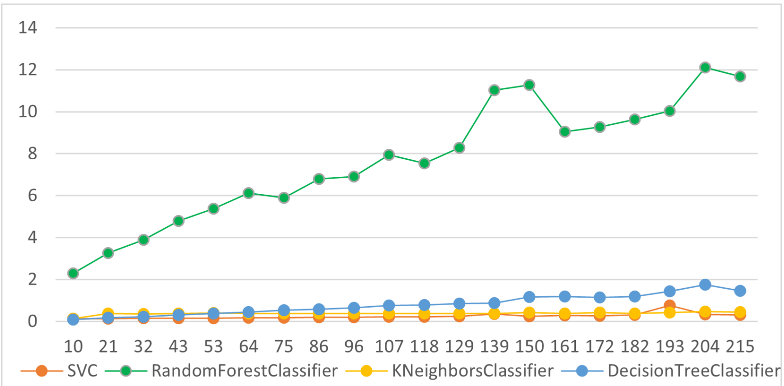
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the SVD method for Malgenome dataset

Best results

	MALGENOME SVD							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	5%	0,9677	30%	0,9829	5%	0,9732	45%	0,9865
Recall	5%	0,9653	30%	0.995	5%	0.9876	45%	0.995
Precision	5%	0,9701	30%	0,971	5%	0,9591	45%	0,9781
Accuracy	5%	0,9473	30%	0,9716	5%	0,9554	45%	0,9777

Discussion:

The best results with SVD are observed between 5% and 45% of attributs:

DecisionTreeClassifier:

With 10 attrubites 5% among 215, we obtained an F1-Score 96.77% and Accuracy 94.73%

KNeighborsClassifier

With 64 attrubites 30% among 215, we obtained an F1-Score 98.29% and Accuracy 97.16%

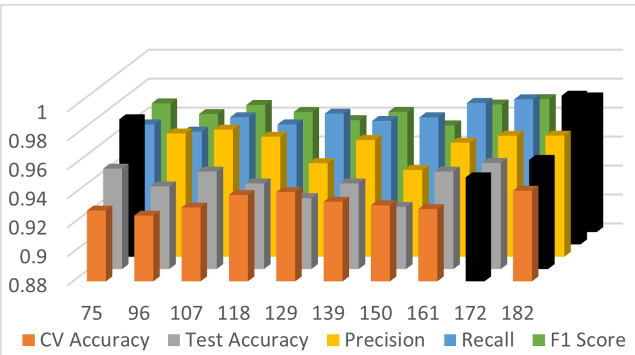
RandomForestClassifier

With 10 attrubites 5% among 215, we obtained an F1-Score 97.32% and Accuracy 95.54%

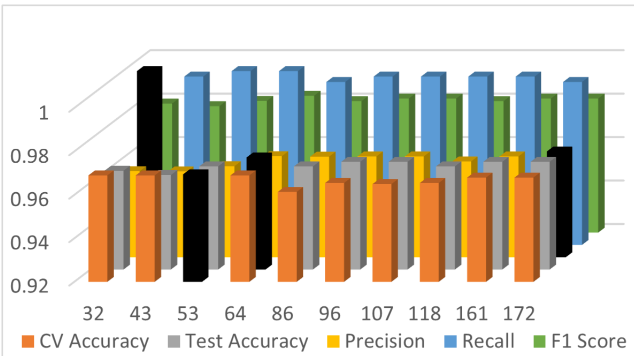
SVC

With 96 attrubites 45% among 215, we obtained an F1-Score 98.65% and Accuracy 97.77%

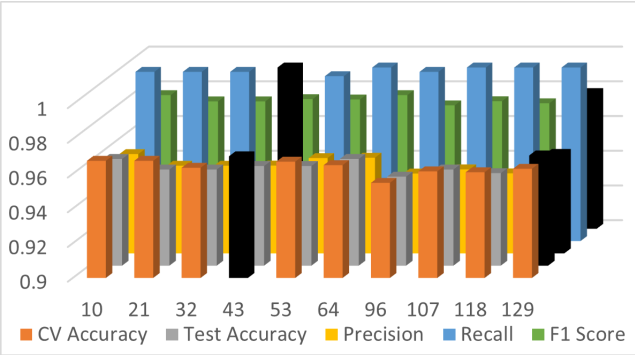
Method 03: Independent Component Analysis (ICA)



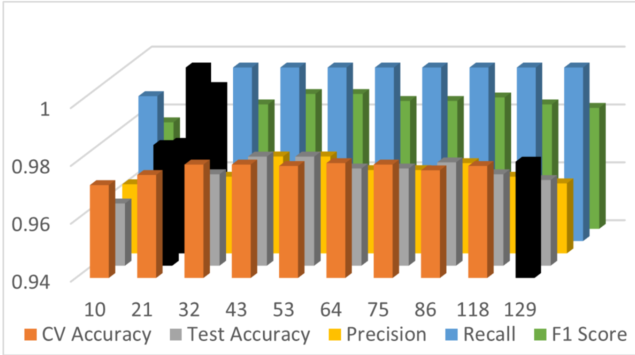
G1: Decision Tree Classifier



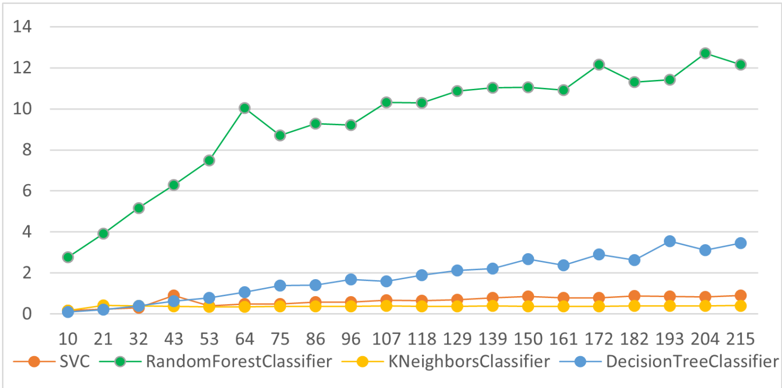
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the ICA method for Malgenome dataset

Best results

	MALGENOME ICA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	85%	0,973	30%	0,983	60%	0,9782	10%	0,989
Recall	85%	0,9827	30%	1	60%	1	10%	1
Precision	85%	0.9636	30%	0.9665	60%	0,9573	10%	0,9782
Accuracy	85%	0,9554	30%	0,9716	60%	0,9635	10%	0,9817

Discussion:

ICA has also improved the performance of the different classifier. we have obtained :

DecisionTreeClassifier:

With 19 attrubites 85% among 215, we obtained an F1-Score 95.73% and Accuracy 95.54%

KNeighborsClassifier

With 25 attrubites 30% among 215, we obtained an F1-Score 98.3% and Accuracy 97.16%

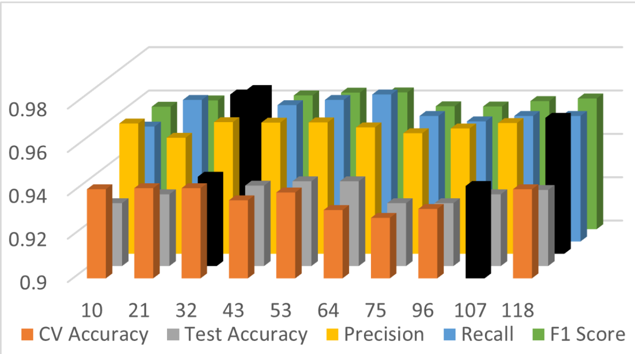
RandomForestClassifier

With 38 attrubites 60% among 215, we obtained an F1-Score 97.82% and Accuracy 96.35%

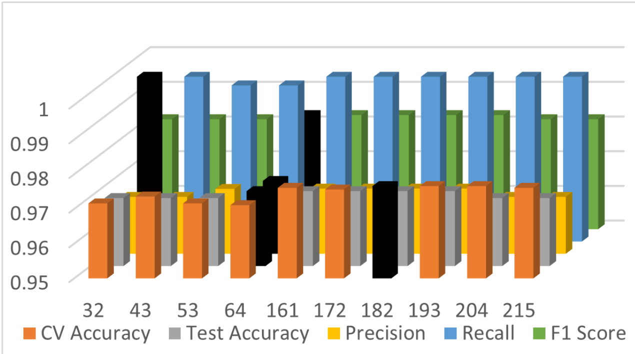
SVC

With 70 attrubites 15% among 215, we obtained an F1-Score 98.9% and Accuracy 98.17%

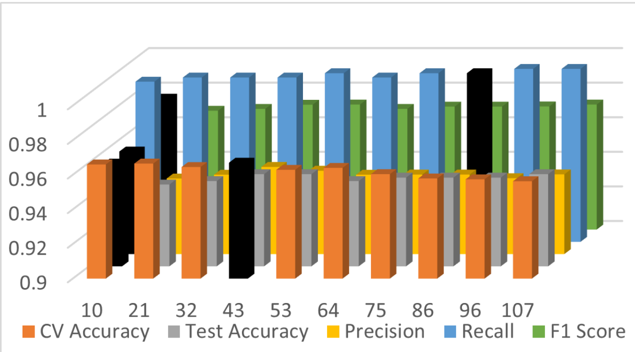
Method 04: Kernel PCA (KPCA)



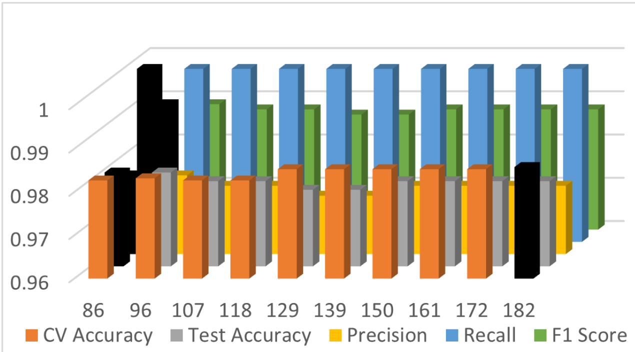
G1: Decision Tree Classifier



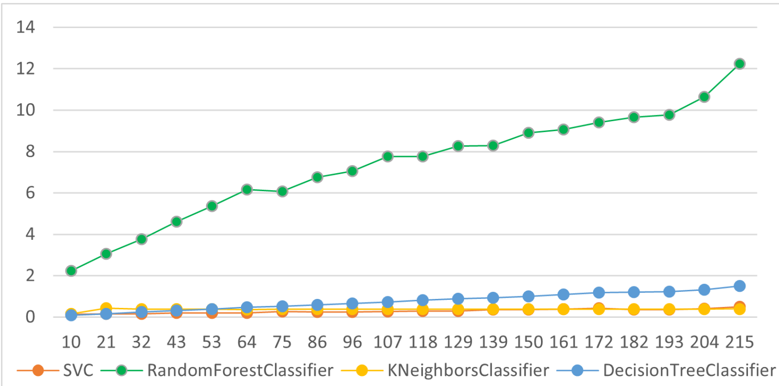
G2: K-Nearest Neighbor Classifier



G3: Random Forest Classifier



G4: SVM Classifier



The result of the KPCA method for Malgenome dataset

Best results

	MALGENOME KPCA							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	15%	0,9642	30%	0,9829	5%	0,9757	40%	0,989
Recall	15%	0,9678	30%	0.995	5%	0.9926	40%	1
Precision	15%	0.9607	30%	0,971	5%	0,9593	40%	0,9782
Accuracy	15%	0,9412	30%	0,9716	5%	0,9594	40%	0,9817

Discussion:
With KPCA we have also noticed an improvement in results:
DecisionTreeClassifier:
With 32 attrubites (15%) among 215, we obtained an F1-Score 96.42% and Accuracy 94.12%
KNeighborsClassifier
With 64 attrubites (30%) among 215, we obtained an F1-Score 98.29% and Accuracy 97.16%
RandomForestClassifier
With 10 attrubites (5%) among 215, we obtained an F1-Score 97.57% and Accuracy 95.94%
SVC
With 86 attrubites (40%) among 215, we obtained an F1-Score 98.9% and Accuracy 98.17%

Synthesis
the best method according to accuracy, F1 score, and low percentage is Kernel PCA (KPCA)
F1 Score: 98.9%
Accuracy: 98.17%

Second-Best Method is Independent Component Analysis (ICA) with:
Accuracy: 97.97%
F1 Score: 98.77%

III.7 Method proposal

The ICA and KPCA methods have given the best results the next step aims to test the effect of merging two dimension reduction methods

1. Extraction / Extraction:
   In this phase we will do a hypridation of the ICA and KPCA method

according to the architecture illustrated in figure III.31

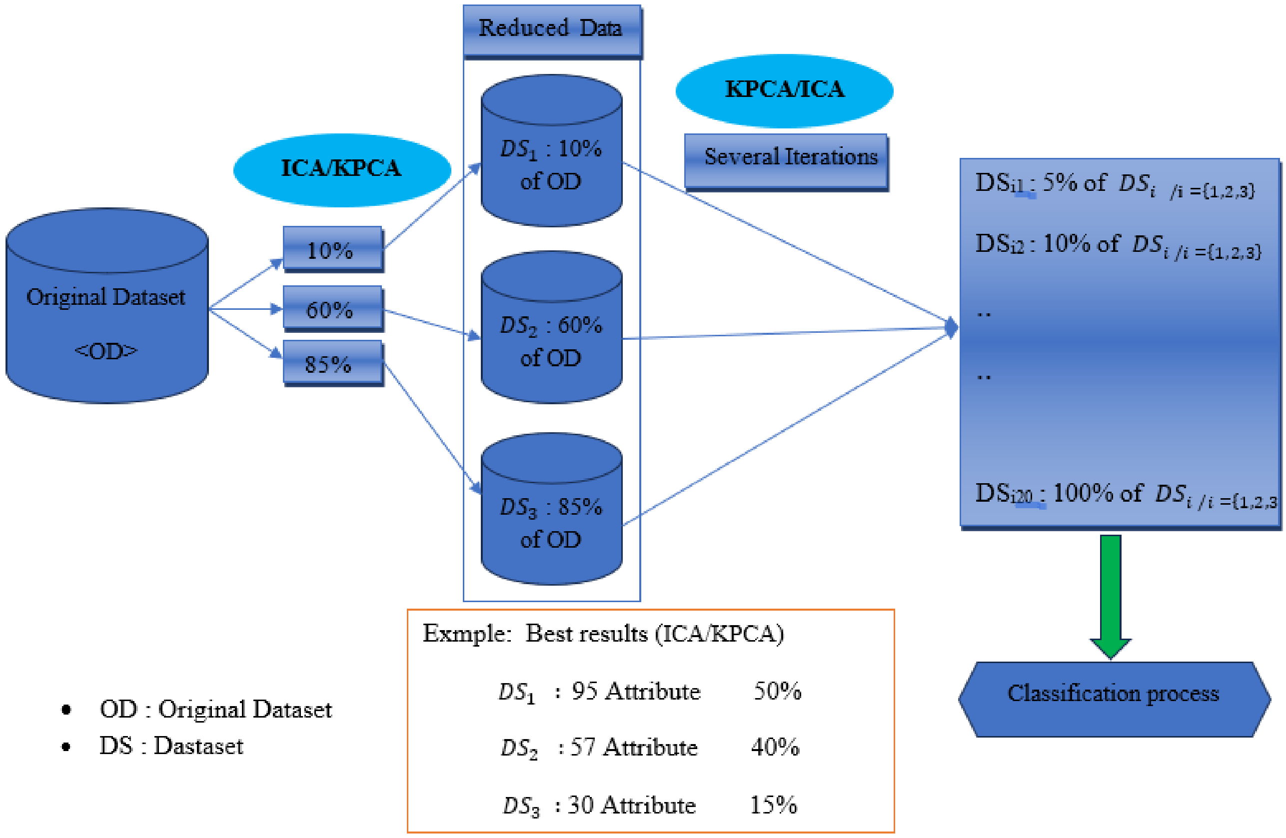


Figure III.31: proposed method architecture: ICA-KPCA with Best

2. Extraction/Selection
   Selection / Extraction
   this step aims to marge the ICA method with Information Mutuelle (IM) selection method which is considered as the best selection method according to the results obtained in [60],(Figures III.32

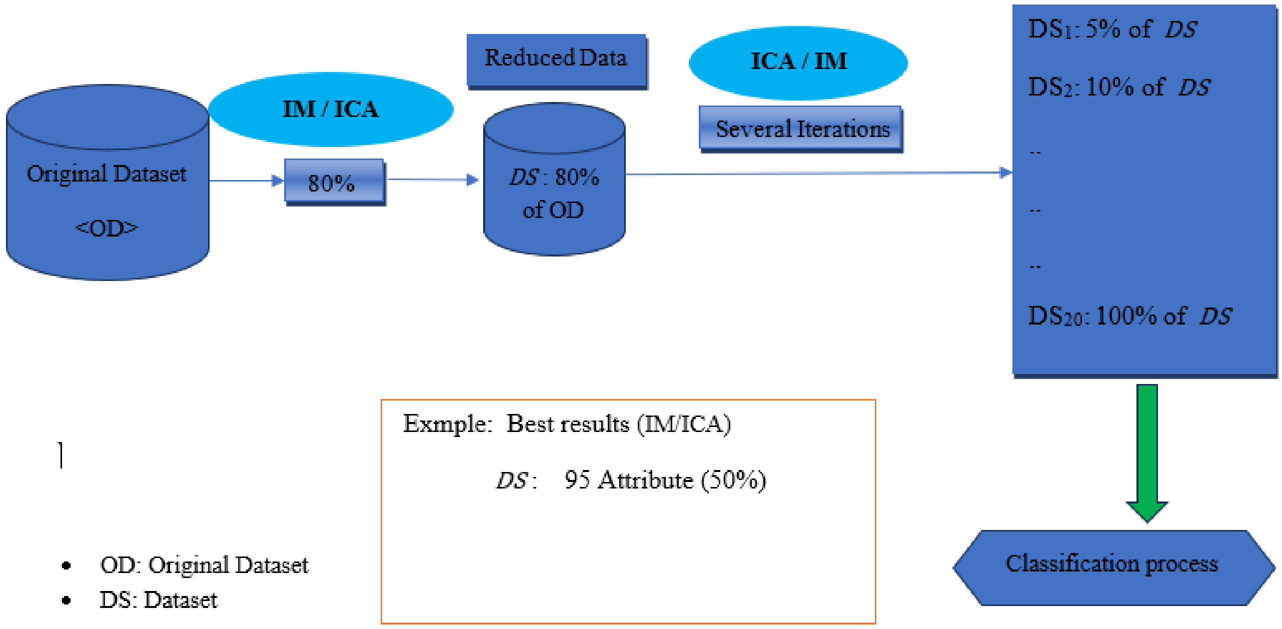


Figure III.32: proposed method architecture: IM-ICA with Best

Best results
After combining different methods around 21 times, we arrived at this KPCA 60% followed by ICA on malgenome dataset is the best resultants

	MALGENOME KPC60							
	DecisionTreeClassifier		KNeighborsClassifier		RandomForestClassifier		SVC	
F1 Score	95%	0,9766	55%	0,9829	35%	0,9794	35%	0,9878
Recall	95%	0,9802	55%	0,9975	35%	1	35%	1
Precision	95%	0,973	55%	0,9688	35%	0,9596	35%	0,9758
Accuracy	95%	0,9615	55%	0,9716	35%	0,9655	35%	0,9797

Discussion:

In the defferent hybrdation we observed a decrease in classifier performance classifiers performance compared to the individual methods we have also noted that feature extraction better than feature selection method.
the feature extraction method have imporved the model perfermence without lossing important informtion

III.8 Application overview

The “Feature Extractor” application provides a comprehensive platform for analyzing and processing the dataset. It features an easy-to-use user interface with two primary tabs: “Feature Extraction Application” and “About”.

Tab 1: The application interface titled “Feature Extraction Application” facilitates dataset selection, feature percentage specification, classifier selection, feature extraction method selection, and data selection. Users can browse and select datasets, define range and augment feature percentages, choose classifiers including DecisionTreeClassifier, KNeighborsClassifier, RandomForestClassifier, and SVC, select feature extraction methods such as PCA, SVD, ICA, and Kernel PCA, and specify sample data size relative to hybridization methods. The layout is organized with labels, input fields, checkboxes, radio buttons, and a “Run” button to perform the feature extraction process. The footer displays copyright information for the application.

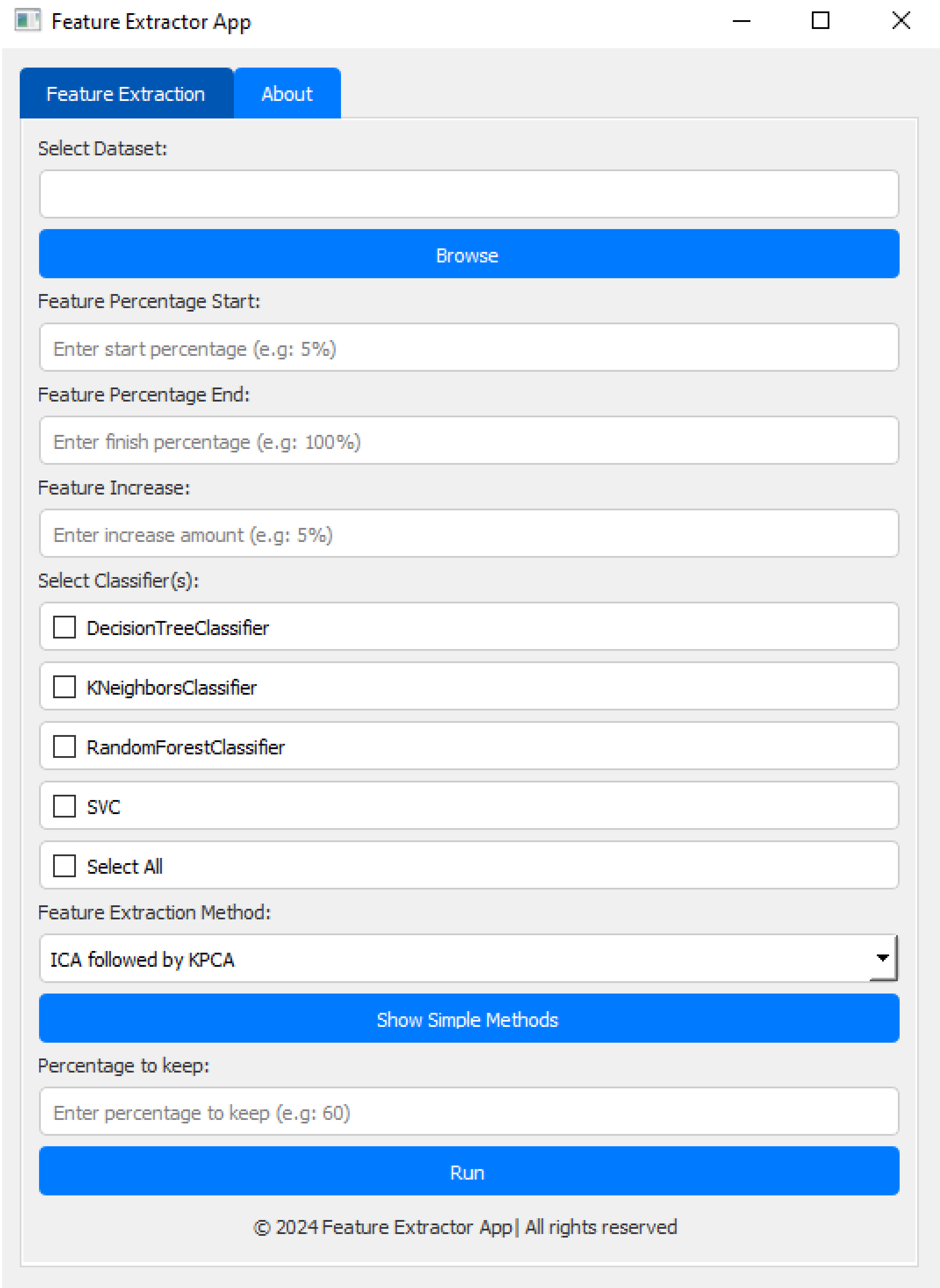


Figure III.33: Interface Application "Feature Extractor App" Tab

Tab 2: titled "About" provides an overview of the application's features and functionalities. It describes how users can utilize the app for dataset processing, feature extraction, classification, and result saving. Key features highlighted include dataset selection via browsing, flexible feature selection parameters such as start percentage, end percentage, and increment, a variety of classification algorithms to choose from, feature extraction methods including PCA, SVD, ICA, and KPCA, options for data selection including using all data or specifying a sample volume, and the simplicity of initiating the feature extraction and classification process with the "Run" button. Additionally, users are informed about the capability to save analysis results as an HTML file for easy access and sharing. The footer displays the version information and copyright details of the application.

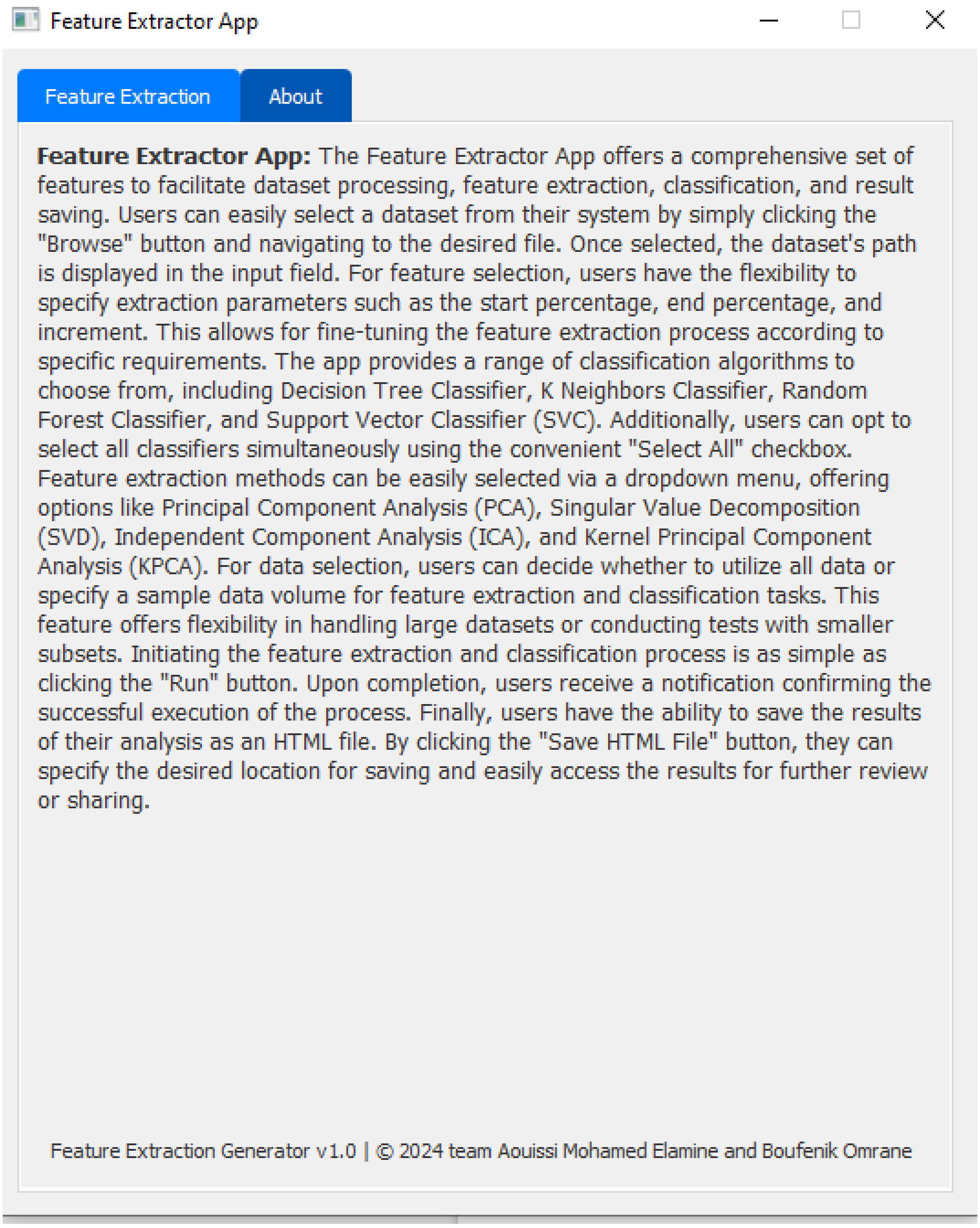


Figure III.34: Interface Application "About" Tab

III.9 Conclusion

In this chapter, we undertook a comprehensive exploration of feature extraction methods and classification algorithms for Android malware detection. We meticulously tested various dimensionality reduction techniques, including PCA, ICA, KPCA, and SVD, to discern their effectiveness in extracting relevant attributes from datasets. Subsequently, we evaluated multiple classification algorithms, such as decision trees, k-nearest neighbors, support vector machines, and random forests, to gauge their performance in classifying malware instances. Following these analyses, we pursued a hybridization approach, combining two feature extraction methods to leverage their respective strengths and enhance detection capabilities. The results of these hybridization experiments provided valuable insights into the synergistic effects of integrating multiple techniques.
Lastly, we provided a detailed overview of the application interface, ensuring clarity and ease of use for users interacting with our system.
Overall, this chapter serves as a pivotal component of our research, laying the groundwork for subsequent chapters while offering valuable findings and methodologies to advance the field of Android malware detection.

General conclusion

In this thesis, we deeply investigated the issue of feature extraction in Android malware detection, using a variety of techniques including Principal Component Analysis (PCA), Independent Component Analysis (ICA), Kernel Principal Component Analysis (KPCA), Singular Value Decomposition (SVD). Here are the main conclusions from our study:

KPCA has shown superior performance compared to other feature extraction techniques. Its ability to identify and extract the most relevant features was impressive, making it an ideal choice for Android malware detection.

Our experiments with PCA and SVD also provided interesting insights, although these methods did not surpass ICA in terms of overall performance.

By hybrdation ICA and KPCA, we noticed a decrease in classifier performance compared to the individual methods.

By combining the ICA extraction method and the MI selection method, we also noticed a decrease in classifier performance compared to the individual methods

These conclusions have significant implications for developing more effective and reliable malware detection systems on Android platforms. By using feature extraction techniques like ICA. We can enhance user security and mitigate the risks associated with malicious applications.

As for future research perspectives, we plan to apply our individual approach of extraction methods to other datasets to evaluate their generalizability, In addition, we aim to investigate more feature extraction methods and propose new techniques to reduce the number of features while maintaining detection quality. By exploring these avenues, we aim to continually improve the performance and efficiency of malware detection systems on Android devices.

Bibliographie

- [1]: Radoniaina Andriatsimandefitra and Ratsisahanana Caract. Caracterisation et détection de malware Android basses sur les flux d'information. Radoniaina Andriatsimandefitra Ratsisahanana.2015.
- [2]: M. Verleysen and D. François, "The curse of dimensionality in datam mining and time series prediction," in International Work-Conference on Artificial Neural Networks, 2005, pp. 758-770: Springer.
- [3]: Pankaj Gupta, Piyush Kumar, Saksham Wason, and Vishal Yadav. Operating System. 2(2) :37–46, 2014.
- [4]: Solène LIMOUSIN. Android et ios. https://www.supinfo.com/ articles/single/ 9003-android-ios, 2019. Consulted the Janury 29,2020.
- [5]: Umer Farooq. Android Operating System Architecture. Consulted the (July) :2–8, 2018.
- [6]: Intents and intent filters https://developer.android.com/guide /components/intents-filters Consulted the Janury 22,2020.
- [7]: Package. https://developer.android.com/reference/java/lang/Package. Consulted the Janury 22,2020.
- [8]: Long Nguyen-Vu, Jinung Ahn, and Souhwan Jung. Android Fragmentation in Malware Detection. Computers and Security, 87 :101573, 2019.
- [9]: Bernard Lebel. Analyse de maliciels sur android par l'analyse de la memoire vive. https://corpus.ulaval.ca/jspui/bitstream/20.500.11794 /29851/1/34353.pdf, 2018. Consulted the Janury 27,2020
- [10]: Jonathan TELLIER À L'Obtention D E La. Technologies de l'information m. ing. La gestion de l' identité fédérée et hiérarchique pour le paradigme. 22 DÉCEMBRE 2011
- [11]: Franklin Tchakounte. A Malware Detection System for Android. Journal of Chemical Information and Modeling, 53(9) :1689–1699, 2015.
- [12]: Dina Saif, S. M. El-Gokhy, and E. Sallam. Deep Belief Networks-based framework for malware detection in Android systems. Alexandria Engineering Journal, 57(4) :4049–4057, 2018
- [13]: Abdelmonim Naway and Yuancheng Li. International Journal of Computer Science and Mobile Computing A Review on The Use of Deep Learning in Android Malware Detection. International Journal of Computer Science and Mobile Computing, 7(12) :42–58, 2018.

- [14]: Cédric Bertrand. Les malwares sous Android. 2012.
- [15]: Meilleures applications de pare-feu android pour une meilleure sécurité internet sur les téléphones. https://www.digitalprivatevault.com/blogs/best-android-firewall-apps-for-better-internet-security-on-phones, 2019. Consulted the Mars 02,2020.
- [16]: Martin Borek, Gideon Creech, and Unsw Canberra. Intrusion Detection System for Android: Linux kernel system calls analysis. 2017.
- [17]: What is malware analysis? defining and outlining the process of malware analysis. https://enterprise.comodo.com/blog/what-is-malware-analysis/, 2018. Consulted the Juin 06,2020.
- [18]:M. Verleysen and D. François, ”The curse of dimensionality in datam mining and time series prediction,” in International Work-Conference June 2005
- [19]: Ngoc Bich Dao. Réduction de dimension de sac de mots visuels grâce à l'analyse formelle de concepts.Traitement des images [eess.IV]. Université de La Rochelle, 2017. Français. NNT: 2017LAROS010.tel-01753800
- [20]: Guérif, S. (2006). Réduction de dimension en apprentissage numérique non supervisée. PhD thesis, Université Paris 13. [-]
- [21]: H. Hotelling. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, vol. 24, no. 6, pages 417–441,1933.
- [22]: J. R. Quinlan. Induction of Decision Trees. Machine Learning, vol. 1,no. 1, pages 81–106, 1986.
- [23]: D. L. Padmaja and B. Vishnuvardhan, ”Comparative study of feature subset selection methods for dimensionality reduction on scientific data,” in 2016 IEEE 6th International Conference on Advanced Computing (IACC), 2016, pp. 31-34: IEEE.
- [24]: A. S. Eesa, Z. Orman, and A. M. A. Brifcani, ”A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems,” Expert Systems with Applications, vol. 42, no. 5, pp. 2670-2679, 2015.
- [25]: P. Jindal and D. Kumar, ”A review on dimensionality reduction techniques,” International journal of computer applications, vol. 173, no. 2, pp. 42-46, 2017.

- [26]: Jiliang Tang, Salem Alelyani and Huan Liu, “Feature Selection for Classification: A Review”, In Data classification: Algorithms and Applications, pp. 1-37, 2014.
- [27]: U. M. Khaire and R. Dhanalakshmi, ”Stability of feature selection algorithm: A review,” Journal of King Saud University-Computer and Information Sciences, 2019.
- [28]: M. Ramaswami and R. Bhaskaran, “A Study on Feature Selection Techniques in Educational Data Mining”, Journal of Computing, Vol. 1, No. 1, pp. 7-11, 2009.
- [29]: R. Kohavi and G.H. John, “Wrappers for Feature Subset Selection”, Artificial Intelligence, Vol. 97, No. 1-2, pp. 273–324, 1997.
- [30]: Zebari et al. / Journal of Applied Science and Technology Trends Vol. 01, No. 02, pp. 56 –70, (2020)
- [31]: Samina Khalid, Tehmina Khalil and Sharmila Nasreen, “A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning”, In Proceedings of 46 Conference on Science and Information, pp. 372-378, 2014.
- [32]: Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. Feature Selection for High-Dimensional Data; Springer: Berlin/Heidelberg, Germany, 2015.
- [33]: Liu, H.; Setiono, R. Chi2: Feature selection and discretization of numeric attributes. In Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 5–8 November 1995; pp. 388–391. [CrossRef]
- [34]: Urbanowicz, R.J.; Meeker, M.; La Cava,W.; Olson, R.S.; Moore, J.H. Relief-based feature selection: Introduction and review. J.Biomed. Inform. 2018, 85, 189–203. [CrossRef]
- [35]: Nasiri, H.; Alavi, S.A.. A novel framework based on deep learning and ANOVA feature selection method for diagnosis of COVID-19 cases from chest X-ray images. Comput. Intell. Neurosci. 2022, 2022, 4694567. [CrossRef] [PubMed]
- [36]: Singh, B.; Kushwaha, N.; Vyas, O.P. A feature subset selection technique for high dimensional data using symmetric uncertainty. J. Data Anal. Inf. Process. 2014, 2, 95. [CrossRef]

- [37]: https://www.javatpoint.com/feature-selection-techniques-in -machine-learning March 07, 2024
- [38]: https://www.kaggle.com/code/prashant111/comprehensive-guide-on-feature-selection March 11, 2024
- [39]: R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction," J. Appl. Sci. Technol. Trends, vol. 1, no. 2, pp. 56–70, 2020, doi: 10.38094/jastt1224.
- [40]: https://www.snowflake.com/guides/feature-extraction-machine -learning/ February 6, 2024
- [41]: I. T. Jolliffe, Principal Component Analysis, Springer, 2nd ed., 2002.
- [42]: https://medium.com/nerd-for-tech/dimensionalityreduction-techniques-pca-lca-and-svd- f2a56b097f7c February 10, 2024
- [43]: https://medium.com/@manan_7/pca-intuition-behind-it-6629644fee53 March 1, 2024
- [44]: https://blent.ai/blog/a/acp-tout-savoir
- [45]: https://complex-systems-ai.com/analyse-des-donnees/les-techniques-de-reduction-de-dimension/ March 3, 2024
- [46]: Jutten, 1987 Jutten, Chritian. 1987. Calcul neuromimétique et traitement du signal - Analyse en composantes indépendantes. Thèse d'état, Institut National Polytechnique de Grenoble.
- [47]: Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10:1299 1319. [-]
- [48]: https://www.geeksforgeeks.org/isomap-a-non-linear-dimensionality-reduction-technique/ March 5, 2024
- [49]: https://neptune.ai/blog/dimensionality-reduction April 2, 2024
- [50]: Omprakash Sain ,A Review on Dimension Reduction Techniques in Data Mining Vol.9, No.1, 2018
- [51]: https://www.geeksforgeeks.org/difference-between-feature-selection-and-feature-extraction/ April 4, 2024

- [52]: https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653/1?file=10391991 March 4, 2024
- [53]: https://archive.ics.uci.edu/dataset/855/tuandromd+(tezpur+university+android+malware+dataset) March 5, 2024
- [54]: https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_1/5854590?file=10391910 March 6, 2024
- [55]: https://docs.jupyter.org/en/latest/, March 10, 2024
- [56]: https://www.python.org/doc/essays/blurb/ March 10, 2024
- [57]: https://medium.com/@trilogicalshelp/pandas-simplifying-data-manipulation-in-python-affabced5434 April 20, 2024
- [58]: https://www.w3schools.com/python/numpy/numpy_intro.asp March 22, 2024
- [59]: https://matplotlib.org/ March 19, 2024
- [60]: Abdelaziz KEDDARI, Fatma HOUACINE Thèse Détection de maliciels Android: Problème de Sélection d'attributs, Université de Saida, 2022-2023.
- [61]: Abdelhakim Hannousse and Salima Yahiouche, Handling webshell attacks: A systematic mapping and survey, Computers & Security, Vol. 108, pp. 1-26, 2021.

ملخص

أدى النمو المتنوع للبرامج الضارة التي تعمل بنظام اندرويد في السنوات الأخيرة إلى إجراء أبحاث مكثفة في مجال تحليل البرامج الضارة واكتشافها، وتم تطبيق نظريات من مجموعة واسعة من مجالات المعرفة العلمية لحل هذه المشكلة. تم استكشاف الخوارزميات من نموذج التعلم الآلي بشكل خاص، وتم اقتراح العديد من استخراج الميزات مثل (تحليل المكونات الرئيسية، تحليل القيمة المفردة، تحليل المكونات المستقلة) وطرق اختيار الميزات مثل (المعلومات المتبادلة) لتمثيل البرامج الضارة كموجهات ميزات لـ استخدامها في خوارزميات التعلم الآلي. نقدم في هذا البحث مقارنة بين عدة تقنيات لاستخراج الميزات. عند فحص النتائج لوحظ أن تقنيات تقليل الأبعاد لها تأثير إيجابي على أداء التصنيف بشكل عام، وخاصة طرق الاستخلاص الفردية مثل تحليل المكونات المستقلة.

الكلمات المفتاحية: أندرويد، استخلاص الميزات، اختيار الميزات، تحليل المكونات الرئيسية، تحليل القيمة المفردة، تحليل المكونات المستقلة، المعلومات المتبادلة، تقليل الأبعاد، التعلم الآلي.

Abstract

The diverse growth of Android malware in recent years has led to extensive research in the field of malware analysis and detection, and theories from a wide range of scientific knowledge areas have been applied to solve this problem. Algorithms from the machine learning paradigm have been particularly explored, and several feature extraction such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Independent Component Analysis (ICA) ,Kernel PCA (KPCA) and feature selection methods such as Information mutuelle (IM) have been proposed to represent malware as feature vectors for use in machine learning algorithms. In this paper we present a comparison between several feature extraction techniques. When examining the results, it was noted that dimensionality reduction techniques have a positive impact on classification performance in general, especially individual extraction methods such as Independent Component Analysis (ICA), Kernel PCA (KPCA)

Keywords: Android, feature extraction, feature selection, PCA, SVD, ICA, KPCA, IM, dimensionality reduction, machine learning.

Résumé

La croissance diversifiée des logiciels malveillants Android au cours des dernières années a conduit à des recherches approfondies dans le domaine de l'analyse et de la détection des logiciels malveillants, et des théories issues d'un large éventail de domaines de connaissances scientifiques ont été appliquées pour résoudre ce problème. Les algorithmes du paradigme d'apprentissage automatique ont été particulièrement explorés, et plusieurs extractions de fonctionnalités telles que Analyse en composantes principales (PCA), Décomposition en valeurs singulières (SVD), Analyse en composants indépendants (ICA), PCA du noyau (KPCA) et des méthodes de sélection de fonctionnalités telles que as mutuelle Information (MI) ont été proposés pour représenter les logiciels malveillants comme vecteurs de fonctionnalités à utiliser dans les algorithmes d'apprentissage automatique. Dans cet article, nous présentons une comparaison entre plusieurs techniques d’extraction de caractéristiques. Lors de l'examen des résultats, il a été noté que les techniques de réduction de dimensionnalité ont un impact positif sur les performances de classification en général, en particulier les méthodes d'extraction individuelles telles que l'analyse en composants indépendants (ICA), la PCA du noyau (KPCA).

Mots clés: Android, extraction de fonctionnalités, sélection de fonctionnalités, ACP, SVD, ICA, KPCA, MI, réduction de dimensionnalité, apprentissage automatique.