DIFFERENTIATING USERS BASED ON CHANGES IN THE UNDERLYING BLOCK SPACE OF THEIR SMARTPHONES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Eric D. Katz

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Marcus Rogers, Chair

Purdue Polytechnic Institute of Technology, Department of Computer and Information Technology

Dr. Eric Dietz

Purdue Polytechnic Institute of Technology, Department of Computer and Information Technology

Dr. John Springer

Purdue Polytechnic Institute of Technology, Department of Computer and Information Technology

Dr. Jay Doyle

MITRE

Approved by:

Dr. Kathryne A. Newton

Head of the School Graduate Program

This dissertation is dedicated to my family who encouraged and believed in me. You pushed me to succeed even when I had doubts. I could not have done this without you. Mom and dad, you especially made this possible. Thank you.

This also dedicated to my wonderful fiancee, Amanda. You put up with all the hardships that came with being with a doctoral student and moving across country. Throughout that time you kept me sane and inspired me to continue.

ACKNOWLEDGMENTS

There are many people who helped me throughout the process of writing this dissertation. I would like to thank all of you and let you know it is truly appreciated. There are a few in particular I would like to especially thank:

Each of my committee members brought the expertise needed to make writing this possible. Dr. Rogers, Dr. Dietz, Dr. Springer, and Dr. Doyle, thank you. You helped guide my research from a rough idea into a successful study.

Dr. Rogers, you have been with me since I started my path in cyber forensics. While this dissertation represents the culmination of my education, it is not the end of my journey. Thank you for always being there.

Dustin Hillman, you read this dissertation more than anyone else, probably including my committee. Thank you for your advice and guidance. Your perspective provided great insights and sense of humor alleviated some painful moments.

Without Mark Guido this dissertation really would have been impossible. Not only did you come up with the PMF concept, no one else understood how that lambda function worked.

ShaRhonda Mirizzi, your tech editing skills are appreciated. So is your patience and tolerance for painstakingly finding the numerous errors.

PREFACE

Originally I never thought I would pursue a PhD. I went to Purdue to earn a masters degree in cyber forensics. When I started, I thought I would earn the degree and go into industry and make the big bucks that I had heard I could earn. The latter never happened. I earned the M.S., but instead of going to industry, I thought I should continue on and pursue my doctorate. The catalyst that changed my mind was being a teaching assistant for Dr. Rogers. I did something that terrifies most people, I stood in front of a classroom and taught. I enjoyed it.

Somehow during my time at Purdue, I evolved from being a student into an expert in cyber forensics and began passing that knowledge onto others. I became the law enforcement coordinator for the cyber forensics program. I was placed in charge of developing training, assisting in investigations, and advising officers in cyber forensics. All of this lead to me realizing I truly enjoyed what I was doing. I imagined that someday I would like to be a professor. The only thing standing in my way, having to earn a PhD.

I began working on this dissertation long before I knew I was going to be writing it. The initial research came from a project I was working on as an intern for MITRE. There, I was introduced to Periodic Mobile Forensics, PMF for short. PMF is a forensic framework for Android mobile phones that utilizes differential analysis to make multiple forensic images over time. From working on PMF, I learned a lot about Python and how the Android OS actually works. Part of my work involved coordinating the first experiment for PMF at Purdue University.

That experiment showed me some of the capabilities of PMF and the possibilities that could come from it. It was the research into designing a masquerade detection system (MDS) from the data that we collected that inspired the research that culminated in my dissertation. While I value the ability to look at full forensic images, I

also own a phone. Knowing how personal my data is and how easily a forensic image could be abused, I thought there had to be a way to preserve user privacy while still providing the security and forensic capabilities of PMF.

That is when I came up with the research idea for this dissertation. Could the changes recorded by the differential nature of PMF be used to create a MDS that kept all user data private until further investigation was needed? All I had to do to find out was master programming, gain expertise in machine learning, and combine them to make a novel MDS. In short I had an idea that would take the next several years of my life to figure out. During that time, I also started a full-time career with MITRE, moved across the country, and most importantly, asked my girlfriend to marry me.

Luckily she said yes and continued to support me as we moved and started a new life together. Writing a dissertation while working full-time has been a challenge, but it is finally finished. This work required me to combine several different disciplines together. I have done my best to present the most pertinent information in a manner that is comprehensible to anyone who wishes to read. With luck, it is also as enjoyable as it is informative.

The results of this research have a multitude of implications on how security and privacy can work together in the future. This is becoming ever more important in an era of big data analytics and increasing integration of smart devices into daily life. The research presented here will show that security and forensic capabilities do not necessarily require sacrificing personal privacy. I hope that this work will inspire future research as I know full well there is still much to learn.

Thank you for reading,

Eric Katz, PhD

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABBREVIATIONS

AdaBoost – Adaptive Boosting

AMD – Android Malware Dataset

AUC – Area Under Curve

BYOD – Bring Your Own Device

C2 or C&C – Command and Control

EER – Equal Error Rate

EMM – Emergency Mobility Management

FAR – False Acceptance Rate

FOA - Focuser of Attention

FRR – False Rejection Rate

HMM – Hidden Markov Models

IDS – Intrusion Detection System

ITU – International Telecommunications Union

LBFGS – Limited-memory Broyden-Fletcher-Goldfarb-Shanno solver

LDA - Latent Dirichlet Allocation

LAN – Local Area Network

MACE – Modified, Accessed, Created, Entry modified times

MDS – Masquerade Detection System

MLP – Multi-Layer Perceptron

MMS - Multimedia Messaging Service (multimedia texts)

MTTA – Mean Time to Alarm

NFC – Near Field Communication

NIST – National Institute of Standards and Technology

NNIDS – Neural Network Intrusion Detection System

OS – Operating System

OTA – Over-The-Air

PAN – Personal Area Network

PCA – Primary Component Analysis

PHIL – Predictive Heuristic Interface Layer

PHMM – Profile Hidden Markov Model

PII – Personal Identifying Information

PITAC – President's Information Technology Advisory Council

PMF – Periodic Mobile Forensics

RBF – Radial Basis Function

RF – Random Forest

ROC – Receiver Operating Characteristic

SMS - Short Message Service (text messages)

SVM – Support Vector Machine

ABSTRACT

Katz, Eric D. Ph.D., Purdue University, May 2020. Differentiating Users Based on Changes in the Underlying Block Space of Their Smartphones. Major Professor: Dr. Marcus Rogers.

With the growing popularity of using smartphones in business environments, it is increasingly likely that phones will be the target of attacks and sources of evidence in cyber forensic investigations. It will often be important to identify who was using the phone at the time an incident occurred. This can be very difficult as phones are easily misplaced, borrowed, or stolen. Previous research has attempted to find ways to identify computer users based on behavioral analysis. Current research into user profiling requires highly invasive examinations of potentially sensitive user data that the user might not be comfortable with people inspecting or could be against company policy to store. This study developed user profiles based on changes in a mobile phone's underlying block structure. By examining where and when changes occur, a user profile can be developed that is comparable to more traditional intrusion detection models, but without the need to use invasive data sets. These profiles can then be used to determine user masquerading efforts or detect when a compromise has occurred. This study included 35 participants that used Samsung Galaxy S3s for three months. The results of the study show that this method has a high accuracy of classifying a phone's actual sessions correctly when using 2-class models. Results from the 1-class models were not as accurate, but the Sigmoid SVM was able to correctly classify actual user sessions from attack sessions.

1. INTRODUCTION

Smartphones are increasingly an important source of evidence in cyber forensic investigations. When an incident occurs, it is necessary to show who was using the phone. This can be challenging as phones can be infected with malware and are easily misplaced, borrowed, or stolen. To combat this, there has been significant research done utilizing machine learning to create user profiles and detect masqueraders. The research in this paper reveals that current methods are inadequate and are often overly invasive of the user's privacy. The conducted study shows that user privacy can be respected while accurately being able to detect incidents when they occur. The following chapters are laid out to explain the importance of this research, show what has been done in the past, provide the methodology used in this experiment, and provide an analysis of the results.

1.1 Statement of the Problem

Mobile phones are one of the most prolific forms of technology available today. The International Telecommunications Union (ITU) estimated that 96% of the world's population, roughly 6.8 billion people, have a mobile phone (I.T.U., 2013). In the United States, that estimation is increased to 109% of the population (I.T.U., 2013). Comscore's October 2014 report indicated that there are 176 million smartphone subscribers in the US, nearly 73% of all US mobile phone subscriptions (ComScore, 2013). Smartphones have become so prevalent that there are more phones being activated than there are new babies being born on a daily basis (AFP, 2013).

Applications known as apps are one of the most common reasons that people are interested in owning a smartphone. In 2009 Apple created the catchphrase "There's an app for that" to market iPhones and the Apple App Store (Grover, 2013). Since then,

the number of apps available to smartphone users has increased almost exponentially with both the Apple App Store and Google Play Store offering more than 1 million apps each (Kalis, 2014). As the number of apps available has increased, so has the smartphone's integration into its users' daily life. According to a poll from Nielsen, the average person uses 26 different apps and logs over 30 hours each month using them (Nielsen, 2014). The growing use of apps has caused mobile data usage to double from 2012 to 2013 and it is expected to increase by 650% by 2018 (CTIA, 2016). This prediction seems to be on track because in 2017, mobile data usage increased by nearly 400% from 2014 and from 2017 to 2018 increased again by 85% (CTIA, 2019b,a) Smartphones accounted for 56% of connected devices and 76% of all wireless traffic in the United States (CTIA, 2016). A 2012 TIME survey showed that 20% of people check their phone every 10 minutes (Gibbs, 2012). The same survey showed that 1/3 of the respondents started to suffer from anxiety when they were separated from their phone for short periods of time (Gibbs, 2012). No other technology has ever been this integrated into its owner's daily life and habits.

Integration of smartphones into daily life has created the push in recent years for the "Bring Your Own Device" (BYOD) trend. BYOD is a trend where employee desire has caused companies to allow employees to bring their own personal electronics to work for use on company networks, products, and services. It is so popular that in the US 85% of companies and 67% of workers already participate in BYOD (IngramMicroAdvisor, 2016; Wainwright, 2016). Often increased productivity and remote work capability are cited as the most compelling reasons more and more companies are implementing BYOD policies, but this increased productivity comes with a risk. 80% of BYOD is unmanaged and most companies are relying on their employees to protect their personal devices and company assets (IngramMicroAdvisor, 2016; Wainwright, 2016).

This reliance is misplaced though as most users do not practice strong security procedures. Standard security mechanisms such as firewalls, antivirus and encryption are uncommon in smartphones and security patches are less frequent and often delayed

(Ruggiero & Foote, 2011; Snell, 2016). Often users won't even engage in basic security precautions such as using a PIN to lock their phone (Tapellini, 2014). Figure 1.1 shows the percentages of how users secured their phones in 2014. A study from the Pew Research Center shows that in 2016 28% of users still did not use a screen lock (Anderson, 2017). While down from the 36% in 2014, that is still a large number of users that are not even taking basic precautions for securing their phone. From this information, it is clear that current security methods are not working and even if they were, users are not implementing them. This leaves not only personal data at risk but company resources and information as well. To make matters more difficult, per the President's Information Technology Advisory Council's (PITAC) report on cyber security, insider threat incidents are rising at a rapid rate (Benioff & Lazowska, 2005). Not only do companies worry about unauthorized 3rd parties stealing information, they have to worry about legitimate users illegally accessing and/or disseminating it as well.

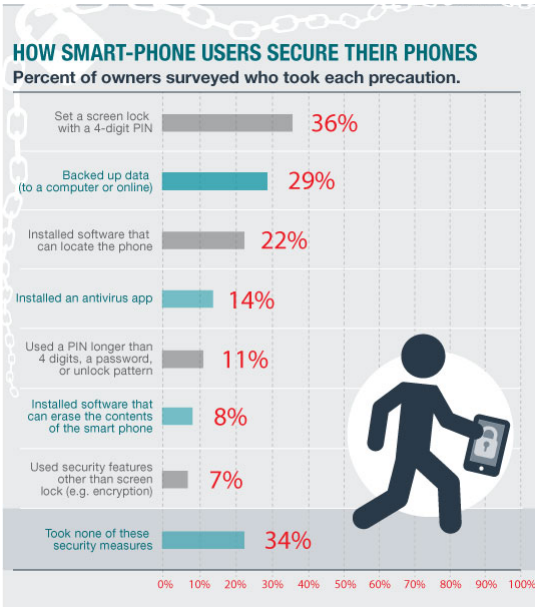

Fig. 1.1. How Smartphone Users Secure Their Phones (Tapellini, 2014)

As phones become more powerful and access more personal and corporate/government systems and information, they will continue to face an increasingly hostile environ-

ment. They will be targeted not only for the physical value they possess, but for the wealth of information stored on them. Phones are now a primary means of social interaction, a method of payment, mobile banking, and have access to private and sensitive information. According to Sophos, the first mobile malware arrived in 2004 and by 2014 has increased to 2,000 new samples per day (Svajcer, 2014). Symantec's 2016 Internet Security Threat Report (ISTR) continues to show this trend by reporting that there was a 214% increase in the number of smartphone vulnerabilities and a 77% increase in the number of viruses for smartphones from 2014 to 2015 (Symantec, 2016). Despite seeing drops in the total amount of mobile malware, Kaspersky reported finding over 5 million malicious packages in 2018 (Chebyshev, 2019). According to Kaspersky's 2014 report 20% of smartphone users experienced a mobile threat that year, while in 2015 the number skyrocketed to an estimate of 25% of smartphone users encountering a threat each month (Garnaeva et al., 2014; CyberEdge, 2015). The 2019 Symantec ISTR reported that one in thirty-six phones has high risk apps installed (Symantec, 2019). To put some perspective on how pervasive a problem this is, there are roughly 7.19 billion people on Earth and roughly 7.22 billion active mobile devices (Zachary, 2014). At that rate approximately 1.8 billion phones encounter some form of cyber threat each month. Malware is not the only threat facing smartphones either. Crime statistics for mobile phones are hard to find, but in 2014, 5.2 million phones were reported lost or stolen in the United States, over 14,000 per day (Deitrick, 2015). A 2017 article from the Mirror reports that over 67,000 phones were stolen in the UK, that breaks down to 183 phones stolen per day (Phillips, 2017). While anti-theft measures are reducing the number of stolen phones, they are clearly still being targeted for theft.

Combining weak security and a high volume of threats means it cannot be assumed that the owner of a smartphone was in control of it when a crime occurs. In 2007 it was estimated that 80-90% of all crimes involved some form of digital evidence (Rogers et al., 2007). It is more important than ever that forensic science be able to identify who or what was actually using the phone at the time in question. Masquerade

detection systems (MDS) are designed to do this by determining between the normal user and an alternative party. They detect anomalies in user behavior predicated on the idea that both normal and malicious behaviors are captured by computers as they are used (Mazhelis & Puuronen, 2007a). These techniques are based on the concept that everyone's cognitive process is unique and that a person's personality, behaviors, and environment reciprocally affect one another leaving a unique impression on the environment (Mazhelis & Puuronen, 2007a). Research has shown that a user leaves a print behind when using a computer and that this print can be detected using machine learning and neural networks. Analysts can then use the print to identify a user in a manner similar to how fingerprints are used to identify individuals (Ryan et al., 1998). This follows Locard's Principle of Exchange, a criminal justice theory, that states; it is impossible for a criminal to act without leaving traces of their presence at the scene of the crime (Handbook, 2012). In the digital world this means any activity taken by the user will leave traces of that activity recorded by the computer system.

Most MDS that have been developed rely on collecting large amounts of system and user data. This can create a conflicting requirement with user privacy needs as sensitive and personal information is often captured and analyzed in order to create the user profile. This is especially true on smartphones which can contain data from all aspect of a person's life.

New methods designed to operate in mobile environments are needed. These methods must be effective, transparent and cannot interfere with how the user experience or the user may disable them. This study proposes a novel method of identifying smartphone owners without the need to analyze sensitive user data. Instead it will monitor resource utilization, specifically it will look at where changes occur in the underlying drive space of a smartphone over time. This will allow user profiles to be built that can distinguish the correct user from a different one without the need to expose personal or sensitive data.

1.2 Scope

This research assessed whether it is possible to develop a distinguishable user profile from collected information indicative of mobile device storage changes. The goal was to establish user profiles and to determine if a different user is utilizing the mobile device. The corpus of data was from the Purdue Masquerading Experiment, where a Purdue team and a MITRE team identified, with significant accuracy, users of the mobile devices other than an enterprise issued device user based solely on collected device behaviors. The data from the Purdue experiment included everything needed to reproduce full forensic images of the phones in the study. Only the data related to when and where changes occurred was used to assess whether similar results to the Purdue Experiment can be achieved utilizing this more primitive dataset.

1.3 Significance

The research proposed here is important for several reasons. The first reason is that being able to forensically attribute who was or was not using a smartphone when an incident occurs is import to any investigations. Smartphones are easily compromised despite storing significant amounts of sensitive information. For this reason, smartphones are the targets of cyber-criminals both physically and digitally. The second reason is because evidence from smartphones is becoming increasingly important in court. Possession of a smartphone is not a reliable means of determining who or what was controlling the phone at a given time. Malware and other users can quickly change how a phone is used without the owner ever knowing the phone is no longer under their control even if it is in their pocket. In the first half of 2109, Kaspersky detected over 1.6 million new malware packages (Chebyshev et al., 2019). This is down in total number from the year before, but only because cyber-criminals are standardizing the malware and not because they are slowing down (Chebyshev et al., 2019). The final reason is because traditional techniques for masquerade and intrusion detection do not work in mobile environments and new methods are needed

that can take advantage of how smartphones function while preserving the privacy of the owner.

As discussed in section 1.1, smartphones are one of the most prolific forms of technology available today. They have unprecedented levels of access to their owner's lives and the networks to which they connect. Unfortunately, smartphones can be easily, lost, stolen, infected, or otherwise compromised. This potentially grants illegitimate third parties access to all of the personal identifying information (PII) and any business or government information stored on that phone. Organizations must also beware of insiders abusing BYOD to steal information. The combination of these means the ability identify who was using the device (forensic attribution) when an incident occurs is becoming even more important.

Establishing forensic attribution has always been difficult for science, but especially so in cyber investigations. The existence of a file or network resource being accessed doesn't by itself prove who was responsible. The internet allows computers to be accessed from anywhere in the world, even while the legitimate user is operating it. There are several cases where evidence seemed to point to one person but investigation revealed a different person or introduced reasonable doubt. For example, in the *U.S. v Moreland (2011)*, Moreland's conviction for possession of child pornography was overturned because the prosecutor could not establish who was accessing the computer when the images were downloaded (U.S., 2011).

Falsely accusing someone with possession of child pornography can ruin their lives, careers, and relationships. While legally innocent until proven guilty, public perception does not care. A study from the University of Oxford Center for Criminology shows that even after being acquitted, the lives of the innocent are negatively impacted for years afterwards (Hoyle et al., 2016). Similar issues can arise with many types of crimes and smartphones are more often being used as a source of evidence. As phones are targeted by more sophisticated adversaries and malware, ownership of the phone cannot be relied upon as means of establishing the identity of a smartphone's user. With lax security, phones are too easily compromised as shown in the

2104 Consumer Reports and 2017 Pew Research studies (Tapellini, 2014; Anderson, 2017). A method of determining who was controlling the phone needs to exist or else the provenance of any evidence from it could be called into question. That is an unacceptable state for any civil or criminal case.

MDS use behavioral characteristics captured by computers to establish user profiles. The behavior of a user for any given session can then be compared to the legitimate profile to determine if they are the same person. A crucial step in designing a MDS is determining what features need monitored and how to model them (Mazhelis & Puuronen, 2005). Much of the research done in Intrusion Detection Systems (IDS) MDS rely on statistical models involving truncated or enriched Unix commands entered by the user in a command line terminal (Dash et al., 2005). This option is not normally available on and is not how smartphones are typically used. Newer methods of masquerade detection need to be developed in order to operate in mobile environments. They need to take advantage of the unique and pervasive nature of smartphones while not interfering with user experience. The research proposed here aims to bridge this gap by exploring a novel means of user identification by monitoring the changes in the underlying drive space of a smartphone. Doing this takes advantage of how smartphones work while still preserving user privacy. This technique is forensically sound and will allow for more in depth forensic investigations to occur when necessary. With the amount of risk smartphones are exposed to, this research has never been more important.

1.4 Research Question

The question proposed for this research was: How accurate can the process of assessing block structure changes of a mobile device during normal use be in detecting:

- masquerading users
- mobile malware

- the exploitation of the device

To answer this question, two hypothesis were tested.

- **Hypothesis 1:** The behavior of the owner of a phone creates unique patterns in the block space of a phone that will allow it to be uniquely identified and classified from other phones of the same make and model.
- **Hypothesis 2:** The behavior of a masquerader (human or software) is different from the owner and will leave patterns in the block structure of a smartphone that can be used to determine when the phone is not being used by the owner.

1.5 Assumptions

The assumptions for this research included:

- Participants were using their smartphones from the Purdue Experiment in the same manner they would use their personal phones
- Participants did not give their phones to other people to use during the Purdue Experiment.
- Some participant behavior is more easily profiled than others.
- The results from this research can be applied to a larger population.
- The block structure of a phone's drive is granular enough system that it can be used to create profiles.
- The sessions times from the original Purdue Experiment (roughly every 24 hours) are granular enough to be useful in creating profiles and testing masquerade attempts.

1.6 Limitations

The limitations for this research included:

- The data for this study came from a prior study by MITRE in conjunction with Purdue University and was not specifically designed for this research.
- There are only 35 subjects in the study.
- The number of sessions per phone is limited with an average of 30.
- The participants might have used their phones in an irregular manner because they were aware it belonged to an experiment
  - IRB approved subterfuge was used to mitigate this
- The study data was originally collected for a masquerading experiment looking a higher lever user behavior pattern.
- The session time is based on the intervals between data collection runs from the smartphone. Scans were run approximately once every 4 hours to detect changes and uploaded roughly once every 24 hours, excluding holidays and weekends
- The study only recruited student and staff from Purdue University as test subjects
- Unencrypted user information was not allowed to be examined or analyzed as part of this research

1.7 Delimitations

The delimitations for this study included:

- The study was only conducted on Samsung Galaxy S3s
- The study only used Android smartphones running Android OS Gingerbread and may not be applicable to non-Android or more recent phones

1.8 Definitions

For the purposes of this dissertation, the following terms are defined as follows:

***Insider threat*:** users with legitimate access to company assets who use that access, whether maliciously or unintentionally, to cause harm to the business (Goldstein, 2019).

***Intrusion Detection System*:** A software application that can be implemented on host operating systems or as network devices to monitor activity that is associated with intrusions or insider misuse, or both (Grance et al., 2002).

***Masquerader*:** is an attacker who succeeds in obtaining a legitimate user's identity and impersonates another user for illegitimate purposes (Salem, 2012).

***Masquerade Attack*:** is a class of attacks, in which a user of a system illegitimately poses as, or assumes the identity of another legitimate user (Salem, 2012).

***Mobile Phone*:** a wireless handheld device that allows users to make and receive calls and to send text messages, among other features (Techopedia, 2020).

***Smartphone*:** For the purpose of this paper smartphone is synonymous with mobile phone.

***SVM*:** Support Vector Machine – a set of supervised learning methods used for classification, regression and outliers detection (Learn, 2020b).

***ocSVM*:** One class SVM - an unsupervised algorithm that learns a decision function for novelty detection: classifying new data as similar or different to the training set (Learn, 2020a)

1.9 Summary

This chapter provided the scope, significance, research question, assumptions, limitations, delimitations, definitions, and other background information for the research

project being proposed. The rest of the paper is organized as follows. Chapter 2 provides a literature review on user profiling and masquerade detection. Chapter 3 lays out the methodology that was followed for the proposed research. Chapter 4 is the analysis of the results from experiment. The conclusions from this research and future research proposals are in chapters 5 and 6 respectively.

2. REVIEW OF RELEVANT LITERATURE

This chapter provides a review of the literature relevant to user profiling and masquerade detection. While there has been significant research conducted into the best methods to identify a particular user during a session, little of it has focused on mobile devices. Additionally none of the research has focused on data as primitive as the where changes occur in the storage device itself. This literature review examines multiple methods of modeling user behavior in order to distinguish one user from another. It also covers the social cognitive theories behind how users can be profiled from artifacts and traces left on a computer. There is also a brief discussion on machine learning and the mathematics the support user profiling and masquerade detection.

2.1 Threats to and from Smartphones

Cybercrime is a rising issue. Across all fronts cybercrime is increasingly committed as attackers grow more sophisticated and more devices are connected to the internet. One of the fastest rising targets for cybercrime are mobile phones. According to Sophos, the first mobile malware for smartphones arrived in 2004 and by 2014 had the increased to 2,000 new samples per day (Vanja, 2014). Symantec's 2016 Internet Security Threat Report continues to show this trend by reporting that there was a 214% increase in the number of smartphone vulnerabilities and a 77% increase in the number of viruses for smartphones from 2014 to 2015 (Symantec, 2016). According to Kapersky's 2014 report 20% of Android users experienced a mobile threat that year, while a year later the number skyrocketed to an estimate of 25% of smartphone users encountering a threat each month (CyberEdge, 2015; Garnaeva et al., 2014).

The 2019 Symantec ISTR reported that one in thirty-six phones has high risk apps installed (Symantec, 2019).

Malware is not the only threat facing smartphones either. In 2014, 5.2 million phones were reported lost or stolen in the United States (Deitrick, 2015). As phones gain more access to personal and corporate/government systems they will continue to face an increasingly hostile environment. They will be targeted for not only the physical value they possess, but for the wealth of information stored on them. Phones are now a primary means of social interaction, provide mobile banking, and have access to private and sensitive information. It's no wonder that smartphone are considered to be IT's weakest security link (CyberEdge, 2015).

Unfortunately, smartphone security has not kept pace with traditional computer security (Ruggiero & Foote, 2011). Standard security mechanisms such as firewalls, antivirus and encryption are not available in smartphones and security patches are less frequent and often delayed (Ruggiero & Foote, 2011; Snell, 2016). Often users won't even engage in basic security precautions such as using a PIN to lock their phone (Tapellini, 2014). Figure 1.1 shows the percentages of how users secure their phones. From this information, it is clear that current security methods are not working and even if they did users are not implementing them.

Failure to implement security is not just a personal problem. It is estimated that a mobile malware breach could cost enterprises $26.4 million dollars annually (Lookout, 2017). This was based on a study of 588 IT and security leaders from Global 2000 companies conducted by the Ponemon Institute and Lookout (Lookout, 2016). This report shows that the average cost of a single infection can cost almost $9,500 for remediation and can skyrocket to $21,000 if employee credentials or corporate data is stolen (Vijayan, 2016). This includes the cost forensics, loss of productivity, theft of information, and cost of brand damage. These numbers are staggering considering that 54% of the respondents admitted to a mobile malware infection within the past two years and another 12% don't know if they are suffering from an infection.

On average, it can take anywhere between 98 to 197 days for a company to detect a breach (Osborne, 2015). This leaves a lot of time for an adversary to steal credentials and access sensitive information. If those numbers seem inflated, they are not. There is a reason mobile malware sells for $5,000 and up to $15,000 in underground markets (Bach, 2015; Kessem, 2016). These are the same prices as PC-based trojan kits and it shows that the value of mobile malware is increasing as more methods of monetizing infections are found (Bach, 2015). It is estimated that the global cost of cybercrime is as high as $400 billion a year and is estimated to grow to $2.1 trillion per year by 2019 (Morgan, 2016). Each year an increasing percentage of these costs come from losses due to compromised smartphones. The lack of security and wealth of data are enticing more and more cyber-criminals to target smartphones.

New methods designed to operate in mobile environments are needed. These methods must be effective and transparent. They cannot interfere with how the user interacts with their mobile phone or the user may choose to disable them. The following section discusses user behavior and how it affects the phone. It is in these affects that machine learning can be used to create a profile of the user.

2.2 Behavior Profiling

This section explains how user behavior can be identified using data recorded by a computer while the user is operating a device. The implication is that the various methods and operations that a user engages in is unique to them. Identifying the correct feature set to extract and study can then be used to develop a profile. Comparing the profile to data from a given session can then be used to determine who was or was not using the computer for that session. The theories behind user profiling are based on social cognitive theory and Locard's Principle of Exchange. Social cognitive theory explains human functioning in terms of triadic reciprocal causation (Bandura, 1986). In this model, behavior, cognitive, personal factors, as well as environmental events all operate as interacting determinants that influence each other bidirection-

ally (Bandura, 1988). Figure 2.1 below is a diagram showing how the triadic system of determinism works. It means that a person's individuality (personality) effects their behavior and their environment while those factors also have an influence on individuality and each other (Mazhelis & Puuronen, 2005). This interaction becomes important when trying to determine a user because it is the behavioral characteristics that are recorded by the computer system (the environment).

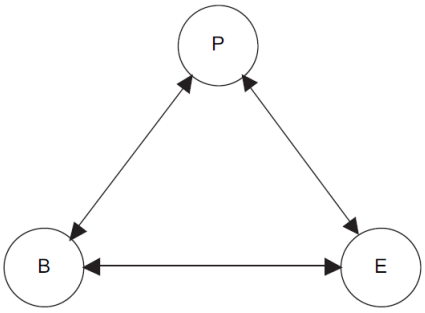

Fig. 2.1. Triadic Reciprocal Determinism (Bandura, 1988)

According to Mazhelis and Puuronen (2007) not all influences have equal importance in revealing individuality being reflected in aspects of behavior in the environment. We are most interested in the effect of behavior on the environment as well as the effect of personality on behavior as shown in Figure 2.2. The relationships the are most influential for user profiling are drawn with solid lines. The reasoning behind this is (Mazhelis & Puuronen, 2007a):

1. **P → B:** That personality will influence behavior, meaning certain actions will be taken and done in an order due to the user's personality

2. **B → E:** Behavior will influence what and how items are changed in the environment

3. **E → B:** The environment itself may impose restrictions on user behavior

By relating the environment to behavior and personality it is possible to uniquely identify a user. Illegitimate users are likely to display markedly different behavior

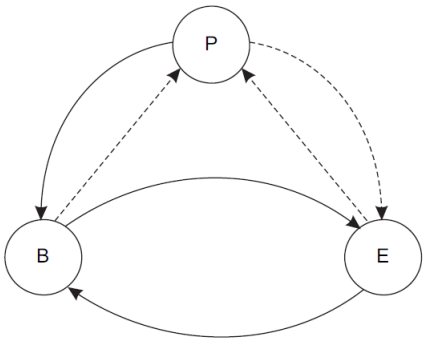

Fig. 2.2. Interesting Relationships (Mazhelis & Puuronen, 2007b)

from legitimate ones (Lunt, 1990). An illegitimate user will lack the same knowledge of the device and its data as the real user. They will also have different goals such as data exfiltration or the theft of personally identifying information (PII) or financial credentials. When applied to this research, comparing the artifacts left in the storage medium from an intruder to a profile of the legitimate user should have enough statistical differences to show that it was not the legitimate user operating the device at the time.

Often when conducting a cyber forensic investigation, the environment (smartphone) is the only thing available for examination. According to Locard's Principle of Exchange whenever a person commits a crime something is left at the scene that was not present when the person arrived (Locard, 1934). While Locard was referring to trace evidence from physical crime scenes, this principle is also true in digital crime scenes. It is almost impossible to avoid leaving a trace in an audit log, a modification of meta data, or an alteration in a sector when using a digital device. These changes are what creates a unique print for each user that can be used in a manner similar to a fingerprint in order to place a user at the scene of a crime (Ryan et al., 1998).

In order to compare this fingerprint, repeated observations of the legitimate user are statistically compared in order to generate a profile. There is a wide variety of different features that can be monitored; including application usage, entered commands, system calls, security events and etc (Salem & Stolfo, 2009). Unfortunately,

users are extremely varied and adapt their behaviors over time. This can make it difficult to identify patterns for them. Human generated data is notoriously noisy, meaning it can be very erratic with several outliers (Lane, 2000). Users constantly change and modify their behavior to accomplish new tasks, they make mistakes, are distracted by interruptions, forget items, and face many other problems (Lane, 2000). In order to compensate for this, a significant amount of research has been done to determine the most effective means of user profiling on computer systems.

The following three sections explore the current literature on user profiling and masquerade detection. They are broken down into different methods of profiling starting with command line attempts as those were the some of the first methods developed. As research advanced, new techniques studying graphical user interfaces (GUI) interfaces emerged. Finally, methods specifically designed to work on cellphones will be examined as they are the focus of the research being proposed.

2.3 Command Line Methods

Automated masquerade detection involves statistical analysis of multiple legitimate user sessions in order to create a profile. A session is the length of time in which the user operated the computer, they can be predetermined or based on when users begin and end interacting with the computer. Profiles are created by monitoring different feature sets that the researcher believes will provide the necessary insights to determine the legitimate user's behavior (Salem & Stolfo, 2009). One of the earliest and most commonly studied feature sets are command line inputs from the user in a UNIX-like environment. This section will briefly cover some of the research and capabilities of this type of study.

Ryan, Lin, and Miikkilainen (1998) were among the first to use neural networks to study masquerade detection. Their NNID (Neural Network Intrusion Detector) used a backpropagation neural network to identify users based on what commands they implemented. Backpropagation essentially causes the estimated answer from

the first pass of a neural network to be fed back through it again until errors are removed (Hecht-Nielsen, 1989). Ryan, et al. (1998) had a limited study of 10 users on one computer that was monitored for 12 days. They examined 100 of the most commonly inputted commands from the subjects to generate profiles by training on eight days' worth of data and testing on the remaining four days' worth of data. Their results showed a remarkable 95% accuracy in detecting anomalous behavior with a false alarm rate of 7% (Ryan et al., 1998). This appears to be near perfect until the limited sample size and highly restricted feature set is considered.

Lane's 2000 thesis on machine learning techniques improved on Ryan et al.'s 1998 work by allowing for all user commands to be included in the model. He also explored the concept of changing user behavior, known as drift, by comparing to possible models, instance-based learning and hidden Markov models (HMMs). Lane's findings showed that users rarely (7%) used the exact same command sequences (Lane, 2000). This means understanding user drift is vital for any intrusion/masquerade detection system. Lane also admitted that he was unable to locate the optimum detection method because the values varied too much amongst individual subjects. Some of the features he tested did show more promise than others though. One of the more important measures was that of the similarity of entered commands to previously entered commands. In the end the best results Lane produces was an 83% detection rate with a false alarm rate of 10% (Lane & Brodley, 2003).

Perhaps one of the most important studies in masquerade detection was done by Shonlau, DuMouchel, Ju, Karr, Theus, and Vardi in 2001. What made this study so important is that Schonlau et al. created a data set of 15,000 commands from 70 different users (Schonlau et al., 2001). This dataset, known as the SEA dataset, has become a defacto standard for research in studying masquerade detection. They used 50 subjects as their legitimate users while the remaining 20 subjects had their data injected into the legitimate data sessions as masqueraders. The feature set they used consisted of command and user names. As with Lane (2000), Schonlau et al. tested six detection schemes, but admitted that no single method could sensible serve as the

sole means of intrusion detection (Schonlau et al., 2001). The best results Shonlau et al. achieved were in their uniqueness metric with a 39.4% success rate and a 1.4% false alarm rate (Maxion & Townsend, 2002).

Maxion and Townsend expanded on Shonlau et al.'s work and improved on their results, increasing the success rate by 56% and lowering the false alarms to 1% (Maxion & Townsend, 2002). Their results had a 61.5% detection rate with only 1.3% false positives. Maxion and Townsend used a Naïve Bayes algorithm to achieve these results. Figure 2.3 shows a comparison of their results compared to Schonlau et al.'s. The x-axis represents false alarm rates and the y-axis represents successful detection rates. It might appear as though the Bayes One-Step Markov method has the best results, but when the cost of false alarms is taken into account, it is both less effective and more expensive to use that the Naïve Bayes method (Maxion & Townsend, 2002). Yung conducted a similar experiment in 2004 that reduced the missing alarm rate by 40% from Maxion and Townsend's work (Yung, 2004).

Dash, Reddy, and Pujari also made use of the SEA dataset in their 2005 study that examined groups of commands instead of individual commands. They used a voting algorithm to create episodes from the sets of user commands and a Naïve Bayes algorithm for detecting masqueraders. Their best result was a detection rate of 88% with a false alarm rate of 12% (Dash et al., 2005). Their contribution shows that episode-based detection schemes can be more effective than individual commands.

Where the Naïve Bayes experiments ignored sequence information, Coull and Szymanski (2008) introduced bioinformatics to masquerade detection, bringing the focus to sequence alignment (Huang & Stamp, 2011). They too used the SEA dataset and achieved even better results than Maxion and Townsend. Adapting the Smith-Waterman algorithm for local sequence alignment to user commands, Coull and Szymanski were able to get a 68.6% detection rate with a 1.9% false positive rate (Coull & Szymanski, 2008).

Another effective method of masquerade detection was developed by Posadas, Mex-Perera, Monroy, and Nolazco-Flores in their 2006 study Hybrid Method for

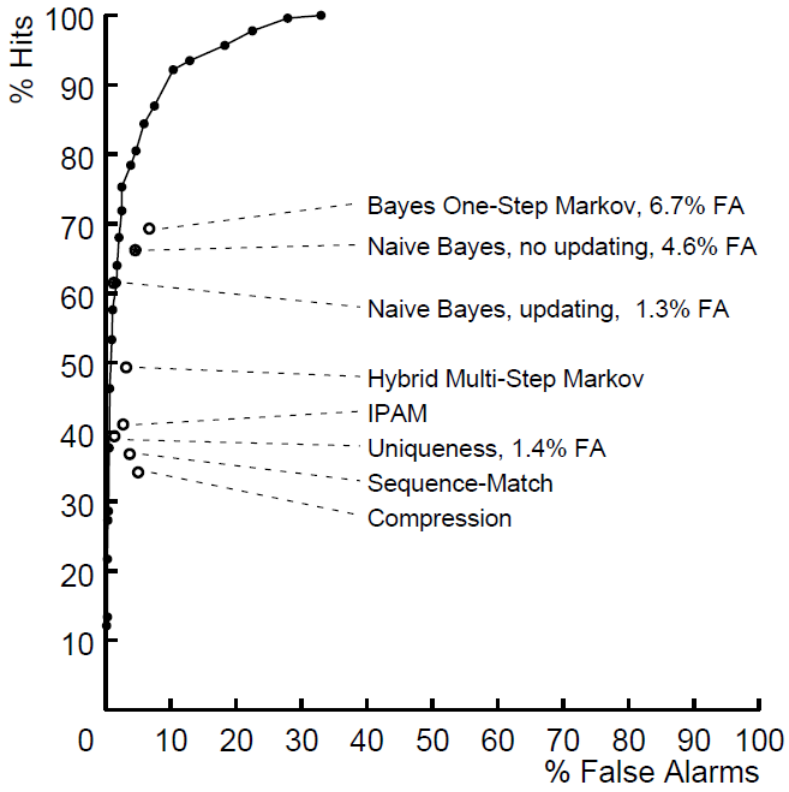

Fig. 2.3. Comparing Maxion and Townsend's Results to Schonlau et al.'s Results (Maxion & Townsend, 2002)

Detecting Masqueraders Using Session Folding and Hidden Markov Models. They used the Sequitur algorithm to create a hierarchical structure for the user commands. Then they applied folding techniques to create compressed versions of user sessions. Their Hybrid-Grammar-HMM was able to reduce both the false alarm rate and the missing alarm rates to about 5% each (Posadas et al., 2006). Their research showed that masquerade detectors do better when based on both frequency and sequence properties of legitimate user profiles (Posadas et al., 2006).

Using a different method of sequence alignment, Profile Hidden Markov Models (PHMMs), Huang and Stamp (2011) were able to show that the number of hidden states have a low impact and two is all that is needed for efficiency. They also conjectured that with the right positional data (session start and end times) that PHMMs

would be more accurate than standard HMMs. Unfortunately, they were using the SEA dataset which did not contain this information. At the time it was created positional data was not considered an important feature, proving that collecting the correct features for profiling is extremely important. To test their theory, Huang and Stamp created a simulated dataset to test PHMMs against and were able to prove that with smaller training sets, PHMMs are better at masquerade detection (Huang & Stamp, 2011).

Despite all of the improvements in masquerade detection via command line profiling several key problems still persist. Often the crossover error rate remained too high to be deployed in working environments to detect masqueraders. Forensically they leave too much error when it comes to identifying if the legitimate user created the session in question. Most importantly though, command line is not how most users interact with a computer. The majority of users today operate in Graphical User Interfaces (GUIs). Smartphones in particular us touch enabled GUIs for most of the user interaction. To detect masqueraders in these environments it is necessary to look at the features that occur in modern operating systems.

2.4 GUI Operating Systems

Even in Unix/Linux environments, commands entered through the terminal are being used less and less. More often than not, users interact with their computer via a Graphical User Interface (GUI). There are a variety of methods researchers have used to profile users in GUI environments. Researchers have studied everything from mouse movement, to text mining, and how files or windows are manipulated. The following subsections cover some of the key research in these areas.

2.4.1 Keyboard and Mouse

In GUIs there are multiple ways for a user to accomplish the same task. Depending on the user's experience and preferences, the manner that they accomplish

an objective can vary. Figure 2.4, seen below, illustrates this concept. Even though there may be a "perfect" or most efficient way to accomplish a task, users will differ in how they achieve it. Profiling how users accomplish these tasks is what allows them to be differentiated from one another.

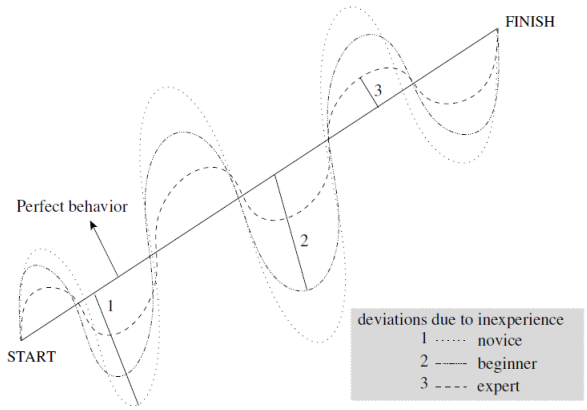

Fig. 2.4. Behavioral Paths for Various Types of Users (Garg et al., 2006)

One of the first studies to take advantage of GUI environments for masquerade detection was conducted by Pusara and Brodley in 2004. They monitored the mouse state as users interacted with their computer. By examining how fast and where a user moved the mouse, clicked, and scrolled the wheel, they were able to successfully distinguish between their 18 subjects (Pusara & Brodley, 2004). Their false negative rate was an acceptable 3.06%; however, several of their users had significantly higher rates of false positives (27.50%), because those subjects did not generate as many mouse events during a session (Pusara & Brodley, 2004). When Pusura et al. applied their smoothing filter to user sessions they were able to lower the false positive rate to 0.43% and the false negative rate to 1.75% (Pusara & Brodley, 2004).

This would be an acceptable rate for testing in a real world environment except that it will not work when the subject is not a prolific user of the mouse. It should be noted that subjects for this study were also restricted to using Windows, Internet Explorer, and browsing the same websites. Opening the restrictions could alter the results of this experiment. In the end, while the results were promising, the experi-

ment showed that mouse events by themselves are not enough to create an effective means of distinguishing users.

Garg, Rahalkar, Upadhyaya, and Kwiat conducted a more flexible and comprehensive study that combined mouse events, typing speed, and system background processes to create 16 different features. They were able to achieve a 92.31% detection rate with only 7.00% false positives using Support Vector Machines (SVMs) (Garg et al., 2006). Unfortunately, they only had three subjects in their test which greatly limits the generalizability of their study. This is an issue with many masquerade detection studies, especially in mobile environments. For numerous reasons, including the protection of sensitive data and subject privacy, there tends to be fewer subjects to make reliable conclusions.

A similar study by Ahmed and Traore in 2007 included additional metrics such as average mouse speed and distance traveled. They ran two experiments, one with twenty-two subjects and another with seven. The smaller experiment was designed specifically to look at some of the confounding factors that were witnessed in the first one. They emphasized the importance of passive monitoring in intrusion detection systems. The dynamics they used were mouse movement, drag-and-drop, and click speed. Ahmed and Traore were able to lower the false acceptance rate (FAR) and false rejection rate (FRR) to 2.46% (Ahmed & Traore, 2007). The Mean Time to Alarm (MTTA) was also calculated for each test. The MTTA is the average time needed to detect a masquerader. Included in this calculation is the time needed to generate a session, the time needed to analyze the session, and the time needed to compare it to the user profile. The shorter the MTTA, the sooner a masquerader can be detected and the appropriate actions can be applied. For Ahmed and Traore's (2007) experiment the MTTA was 13.55 minutes, where only .44 seconds of it was data processing and decision making (Ahmed & Traore, 2007). This is quick and may be reduced further by shortening session time. Further experiments would be necessary to know how that impact the overall detection rate. Their experiment showed

promising results that might be effective in traditional computing environments, but would fail to function properly with smartphones that use touch interfaces.

Bhukya, Kommuru, and Negi (2007) also did a similar experiment looking at mouse and keyboard dynamics together utilizing 18 different features. The events in each session were collected by a logging agent installed on each computer. They compared a one-class SVM to a two-class SVM method. They point out that the one-class SVM was successful 94.88% of the time compared to the two-class SVM's 53.08% (Bhukya et al., 2007). This was only for one user though and other subject's results were not mentioned. There were also only eight subjects, of which results were only shown for four, severely limiting the reliability of their results. The important take-away is that one-class SVMs appear to be not only viable, but might be better for masquerade detection than two-class SVMs.

2.4.2 Navigation

Another aspect of user behavior that has been investigated is that of navigation. How the user access files and resources on a computer can also be captured and used to create a profile. For example, Camina et al. (2011) conducted a preliminary study that looked at how a user browses while working on their own computer (Camina et al., 2011). Their study involved six subjects, three legitimate users and three that acted as masqueraders trying to compromise the legitimate user's computers. They found two interesting results; the first is that it is more difficult to detect masqueraders with users who carry out few tasks or have poor directory organization, the second is that it is more difficult to detect a novice masquerader than an experienced one (Camina et al., 2011). This is likely because the more random the user is (the noisier) the harder it is to profile them. Experienced users are more likely to take specific and directed actions making their behavior more characteristic. This also makes it harder for a masquerader to fake the behavior of a legitimate user. While this study is too

small to provide strong statistical significance, it showed promising results regarding user profiling and masquerade detection.

Gupta conducted an observational study in 2012 that used a keylogger and desktop recording software to capture 13 features about how the computer was used, such as opening a website. Her study had 60 different subjects the first 30 were tested to determine which features would be used and the other 30 subjects had those features monitored as they used their computers. Using cluster analysis she found that a user's sessions will show a strong cohesion (98%) with their other sessions. Meaning that a user's session will be clustered closer to their own sessions than to any other users' sessions (Gupta, 2012). This study comes closer to detecting actual user behavior than any prior. One important aspect of these studies is that they show that recorded system events are not the only means of gathering information on user behavior and other methods can be constructed that might be more successful.

2.4.3 Search Patterns

Search behavior is another aspect of user behavior that can potentially be used to differentiate users. In 2011, Salem and Stolfo conducted a study involving 18 subjects using a computer for four days. They developed a sensor that monitored all registry activity, process creation and destruction, open windows, file access and DLL library activity. In all they had over 10GB of data and more than 500,000 records per user (Salem & Stolfo, 2011). Their hypothesis was that user search behavior is a feature impacted by user intent and thus hard for a masquerader to fake. To test this, they then had a new group of 60 computer science students assigned to three different groups (benign, neutral, and malicious) attempt to masquerade as the user while compromising data. Comparing the masquerade sessions to the legitimate profiles, Salem and Stolfo claim to have 100% masquerade detection with only 1.1% of them being false positives (Salem & Stolfo, 2011). While at face value this seems extremely effective, to conduct this study requires a large database of potentially

sensitive information. The large amount of data was for a fairly small group of users over a short time frame. This is not scalable for large enterprises and highly risky to store such sensitive information. It is also ineffective on mobile devices where that level of aggressive monitoring would devour system resources and have a negative impact on user experience.

2.4.4 Smartphones

Mobile devices are integrating more and more into their user's daily life. Their rapid increase in online applications and services has generated an increased need for security (Shi et al., 2010). The fact that smartphones can be easily lost or stolen means it is possible that a severe security incident concerning private or enterprise data could occur (Mazhelis & Puuronen, 2005). Unfortunately, many masquerade detection systems require the collection of highly intrusive user specific data that is at odds with the security it is trying to accomplish (Samfat & Molva, 1997). Barring that, there is a treasure trove of information on smartphones for masquerade detection researchers that are unavailable on traditional computing devices. This information can include:

- Accelerometer data
- Application databases and usage statistics
- Biometrics
- Bluetooth and USB connections
- Browsing and Search behavior
- Calendars
- Cell Towers
- Contacts

- Email
- EXIF metadata
- GPS
- Call History
- SMS and MMS (Text Messaging)
- WiFi access Points
- etc.

One of the first studies in mobile masquerade detection was done by Samfet and Molva in 1997. Their goal was to detect masqueraders from the cellular provider's perspective. They used call origination, call termination, base station handovers, and location updates to develop user profiles. Using a Wireless Network Simulator (WINES) they simulated user data and traffic patterns. Deviations from the users' call or travel patterns were designed to raise a Focuser of Attention (FOA) alert. While they were never able to deploy their sensors in the real world, their study showed a detection rate between 80 and 100% with less than 5% false alerts (Samfat & Molva, 1997). This limited study was capable of showing when a phone had been stolen and was used to make calls in abnormal locations. It does little to reveal when a masquerader is using a phone in the same area as the legitimate user and does nothing if no calls are made.

Another study looking at mobility was conducted by Sun, Yu, Wu, and Leung in 2004. They simulated user travel paths through a series of cell zones. While they a FAR of less than 3% and a successful detection rate of 80% or more, they admitted that their method was only good for a small subset of users that follow a regular travel itinerary (Sun et al., 2004). Examples of users that are difficult for their method to profile include users that didn't move much, moved in random patterns, or if the phone deviated too far from its normal route (Sun et al., 2004).

A more accurate way of detecting when an illegitimate user has taken a phone was developed by Mantyjarvi, Lindholm, Vildjiounaite, Makela, and Ailisto in 2005. They found that people generate distinct patterns in their accelerometers as they move about (Mantyjarvi et al., 2005). While providing far more accuracy than Samfet et al.'s study (1997), this form of masquerader detection is primary capable of identifying when a phone has been stolen. As with all mobility based detectors, it does little to detect when an illegitimate user picks up a phone and starts rummaging through it to find information or install malware.

Keystroke latency on mobile phones (how fast a user types) was studied by Clarke and Furnell in 2007. They tested both inter-keystroke latency and hold time. Inter-keystroke latency is the time between key presses and hold time is the time it takes to press and release a key. They had mixed results with some users achieving an EER (Equal Error Rate) of less than 2% and others over 20%. The EER is the point where the FAR and FRR are equal. To compare fingerprint recognition has about a 9% EER and hand geometry have 1.5% as can be seen in Table 2.1.

Table 2.1.
Comparing the Performance of Biometric Techniques (Clarke & Furnell, 2007)

Biometric Technique	**EER(%)**	**Company**
Hand Geometry	1.5	Recognition Systems Handkey II
Facial Recognition	2.5	Identix FaceIT
Voice Verification	3.5	OTG SecurPBX
Fingerprint (Chip)	4.5	Infineon VeriTouch
Fingerprint (Chip)	6	Infineon VeriTouch
Facial Recognition	7	Identix FaceIT
Fingerprint (Optical)	9	*Company not disclosed*
Vein	9	Neuscience Biometrics, Veincheck Prototype

The authors admit that the average error rates of their experiments are too high to be considered viable and that confounding factors such as walking could also have increased the error rate (Clarke & Furnell, 2007). They determined that keystroke analysis is not appropriate for all users, specifically those who don't regularly use their phone's keypad (Clarke & Furnell, 2007). The research proposed in this paper is more universal than their research and allows for any type of user to be profiled without requiring specific behaviors from that user. Clarke and Furnell then discussed what would be needed to make an effective authentication mechanism. The requirements include (Clarke & Furnell, 2007):

- Increasing security beyond secret knowledge based approaches (PIN and Password).
- Provide transparent authentication of the user to remove user inconvenience.
- Provide continuous or periodic authentication of the user.
- Function across a range of mobile devices despite hardware, OS, and network differences.

To detect user substitutions, Mazhelis and Puuronen (2007) developed a framework for detectable behaviors and environmental aspects. Characteristics and measures to represent them were also created. Their frame work includes the following assumptions (Mazhelis & Puuronen, 2007b):

- There exists only legitimate user.
- User behavior and environment includes aspects peculiar to the user.
- Operations of the mobile device are managed by the user who is assumed to carry the device and control it through a predefined interface.
- The user accepts monitoring.

- Within a limited time frame, particular aspects of user behavior and environment do not change.
- The user does not suffer from mental stress or other condition that dramatically alters behavior.

They also related various behaviors to environmental characteristics as seen in figure 2.5 These characteristics tie user personality to behavior and behavior to how the device records it.

Table 1 – List of distinctive characteristics

Level	Aspect(s)	Characteristics
Personality reflected in behavior		
High	Way of obtaining information	–Device's facilities usage
		–Sequences of actions followed
	Way of communication with others	–Temporal lengths of actions
		–Temporal intervals between actions in a sequence
	Way of performing tasks	–Retrieving contact details from the device's memory vs. entering them ad-hoc
		–Use of shortcuts vs. use of menu
	Movements	–Routes taken
		–Speed of move conditioned on route/time
	Time devoted to work	–Length of work day
	Changes in the behavior	–Changes in behavior
Middle	Concepts used	–Words or phrases used more often
	Speed of comprehension	–Time of reading a unit of textual information
	Decision-making	–Time between incoming event and response conditioned on time of day
	Accuracy	–Accuracy in typing, in menu item selection, etc.
	Disposition towards communication	–Time devoted to communication
Low	Way of writing	–Pressure, direction, acceleration, and length of strokes
	Way of typing	–Temporal characteristics of keystrokes
	Voice	–Statistical characteristics of voice
Personality reflected in environment		
High	Choice of people to be in contact with	–People contacted, conditioned on type of communication, time, etc.
	Choice of places to visit	–Places visited, conditioned on time of day, week, etc.
	Choice of tools	–Set of software installed
	Changes in the choice of environment	–Changes in the choice of environment
Middle	"Being online"	–Time, when the user is online
Low	Choice of screen resolution	–Current screen resolution
	Choice of volume level	–Volume level

Fig. 2.5. List of Distinctive Characteristics (Mazhelis & Puuronen, 2007b)

Combining Mazhelis and Puuronen's assumptions with their distinctive characteristics provides a direct tie to how user behavior and personality can be recorded by smartphones. This supports social cognitive theories and user profiling. To

prove that their ideas are valid, Mazhelis and Puuronen put their framework to use and attempted to find the best combination of schemes to detect a masquerader in 2007. They compared three schemes, mean of estimated probabilities (MP), product combination of probabilities (PP), and modified mean of the estimated probabilities (modMP). The dataset used was from the Context Project that monitored two small groups of phone users for approximately three months. Recorded data included movement between cells, phone profile, application use, idle/active times, use of the charger, bluetooth connections, and communications (calls and SMS/MMS) (Mazhelis & Puuronen, 2007a). Using a t-test, the Wilcoxon signed ranks test, and the sign test they compared the three schemes using between two and five different features. They determined the best results came from two and three features using the modMP scheme. This study does not state which features were the best and why, nor were their results 100% accurate. Some confounding factors include users changing their routes or configuration shifts in the GSM network.

In a slightly different twist on user profiling, Shi, Niu, Jakobsson, and Chow (2010) developed a method of conducting implicit authentication. They scored a user's behavior based on number of known good events vs number of bad events over time. Examples of good events included calling a known number and visiting a known website. Bad events included calling unknown numbers or visiting new websites. Location information was used as well were being in an area normally visited was weighted good and new areas bad. Time provided a decaying factor so that good events would decay and be weighted less the longer the delay between them. After passing a predetermined threshold, the system would force the user to reauthenticate (Shi et al., 2010). In their experiment, they attempted to test different adversarial models based on an uninformed and informed intruder. For the uninformed intruder, they spliced another user's session into the legitimate sessions as is often done in masquerade detection experiments. For the informed intruder, they combined elements from the legitimate user with the illegitimate one to create a new session.

Shi et al.'s system could detect an intruder 95% of the time after a maximum of 16 interactions with the phone (Shi et al., 2010). This leaves several problems that can be undetected such as an informed intruder accessing specific data and leaving or installing malware on the phone instead of directly using it. The malware in this case would eventually trick the system as its actions would be mixed with good user behavior and authentications, eventually becoming a signifier of "good" behavior. While they had a good idea, there is significant room for improvement.

A 2014 study in intrusion detection on smartphones focused on location. This study used two datasets, Reality Mining and Geolife. Reality Mining ran for nine months and had over 100 users and collected data such as; call logs, event logs, area IDs, and tower IDs. Geolife by comparison tracked 178 users for four years logging GPS data every 1-5 seconds (Yazji et al., 2014). Yazji et al. compared two models using empirical cumulative probability and the Markov transition property. For the empirical cumulative probability, a table was created for each user where the columns were every recorded location and the rows were times. Each cell contained the probability that a user was at that location at any given time. This method could detect an intruder within 15 minutes 94.4% of the time and 92% within five minutes (Yazji et al., 2014). The Markov transition model created profiles using a 3-dimensional table representing location, time, and move where move is the probability of a user moving from their current location to a new one. This model had a 90.5 to 96% chance of detecting an intruder within 15 minutes (Yazji et al., 2014). Overall the results were similar but method two was more accurate. The standard deviations were smaller in model two but the space and computational requirements were greater. The authors therefore recommend model one over model two.

With either model the MDS is storing a large database of where the user was and at what times. Other methods as discussed also stored large databases of sensitive user data such as location, call logs, message, browsing history, biometrics, contact lists, keystrokes, and search history to name a few. In these instances, the MDS' database is as dangerous or more of a security risk than each individual phone that

it is supposed to be protecting. A breach of this database could be catastrophic for both the enterprise and the individual users. All the previous studies also fail to create a forensically useful database in the process. This means breaches cannot be forensically examined to prosecute criminals or improve detection methods. The Periodic Mobile Forensics (PMF) project from MITRE provides both security and forensic soundness. The next section describes PMF and how it functions.

2.5 Periodic Mobile Forensics (PMF)

Periodic Mobile Forensics (PMF) was a research project conducted by the MITRE corporation as part of its MITRE Innovation Program. The concept behind PMF was to use traditional forensic techniques in mobile enterprise environments for a variety of use cases including; insider threat monitoring, malware, masquerade detection, and forensic investigations. This is accomplished by doing differential analysis on the mobile phone, which can be seen in figure 2.6. A small program is installed on the phone in the system partition allowing the original state of the smartphone to be forensically imaged and stored in a database in a secure center (Guido et al., 2016b). The same program then periodically checks the phone for sectors that have changed, in this instance about once every four hours. At set intervals, approximately once every 24 hours for the Purdue Experiment, the changed sectors are then encrypted, uploaded, and stored in the same database as the original image. This allows forensic investigators to insert the changed sectors into the original image, creating a snapshot of the phone at the time the collection was uploaded. For forensic investigations, this means that any altered or deleted data can be recovered. Allowing the examiner to study changes to the phone as they occurred over time can provide a better timeline and reveal evidence that would otherwise not exist.

For masquerade detection, it also means that changes in user behavior can be monitored over time. All of this is accomplished without consuming large amounts of

battery and processing power making for a negligible impact on the user's interaction with the phone. More about how PMF functions can be found in Chapter 3.

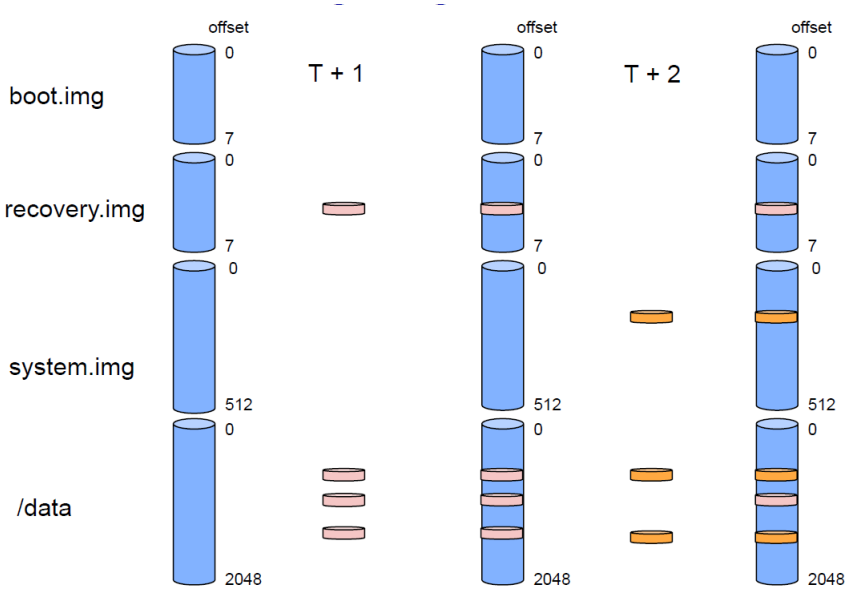

Fig. 2.6. Reconstruction of Images (Guido et al., 2013)

In 2013 in collaboration with Purdue University, MITRE ran an experiment to collect live data from 38 Samsung Galaxy S3's. After achieving approval from both the MITRE and Purdue Institutional Review Boards (IRB), students were given phones with unlimited data, talk, and text plans to use for approximately three months (109 days). The students were told that they were participating in an app usage study to avoid altering their normal behavior. After the study the students were informed of the real purpose of the study and 35 of them signed the post-study consent form to allow their data to be used. This created a database with 35 phones with 872 snap shots of those phones. Most of the periodic images were less than 2GB in size and only one was 11GB, a 50x storage reduction. The database ended up being only 325GB, if left in raw format this database would require roughly 17TB of space which would make it unusable at scale in an enterprise environment (Grover, 2013).

MITRE researchers could reconstruct 821 forensic images of the phones and extract over 1.1 million audit events using 22 different forensic processes (Grover, 2015).

Some of the features included emails, calls, SMS, number of words in SMS, what time was the phone used, and more (Guido et al., 2013). They used the following machine learning techniques to determine which functioned best for creating user profiles and detecting masqueraders (Guido et al., 2016a):

- Latent Dirichlet Allocation
- Decision Tree
- Naïve Bayes
- Linear Support Vector Machine (SVM)
- Radial Basis Function (RBF) kernel SVM
- Adaptive Boosting (AdaBoost)
- Random Forests
- K Nearest Neighbors

Using two pairing methods, one that compares a session to see if it looks more like user X or user Y and the other method looks to see if a session is from phone X or Not-X. Using receiver operating characteristic (ROC) curves to compare the area under curve (AUC) for the different methods, Guido et al. could determine that Random Forests worked best for their feature sets as seen in figure 2.7. The most effective features studied were phone calls, Short message service/multimedia message service (SMS/MMS) messages, and emails. The best results came from the Random Forest algorithm with a median score of 84% across all pairs (Guido et al., 2016a). It is possible that with a less random pairing method that this performance could be improved.

While the PMF project could successfully detect masqueraders using the 22 features, it is also computationally expensive. PMF was designed from the beginning

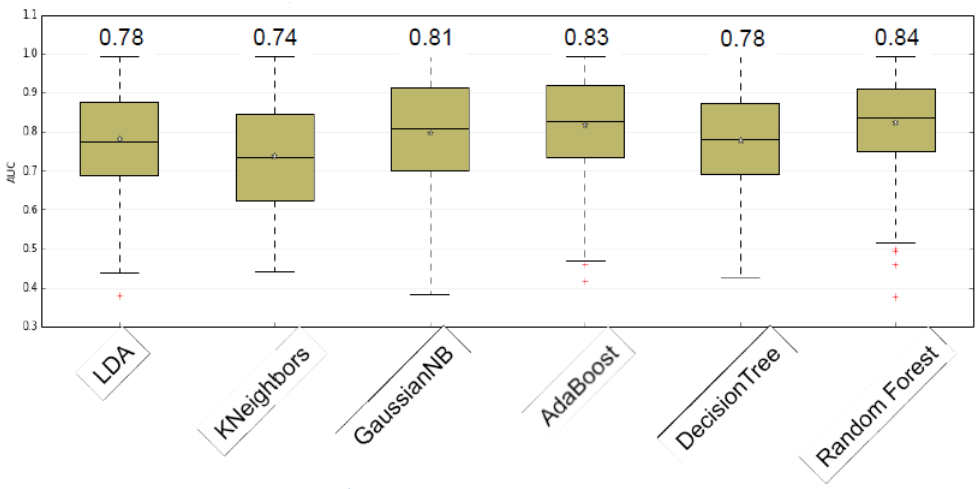

Fig. 2.7. Results from the PMF Project (Guido et al., 2016a)

to be secure and protect user privacy but still can potentially expose users to unnecessary risk. For Guido et al. (2016) to test a user profile, the user data must first be unencrypted and the forensic image of the phone rebuilt. Then the feature sets need to be extracted from each image and stored in a new database. Once the features are extracted the data can be used to build a profile and finally have that profile compared to another session of data to see if it is the legitimate user or not. Even though the databases are stored in a secure environment, if the database was breached all the users' data could be stolen. In an enterprise environment, it also does not't make sense to rebuild every single user's phone every time a collection is made.

2.6 Summary

One thing that all the masquerade detection methods have in common is that they all rely on user behavior leaving changes in the storage medium of a device, as Ryan et al., put it, a "print" on the system (Ryan et al., 1998). Many of the systems designed to collect user information are invasive and can violate user privacy or company policy. The data collection processes can impair user interaction, eat large amounts of system

space, and are computationally expensive thus prohibitive on mobile devices. The database being collected and analyzed itself becomes a prime target for theft and exploitation. Better methods of masquerade detection need to be developed for mobile phones that are transparent to the user and provide security without exposing user privacy unnecessarily. The next chapter provides the framework and methodology for such a method that will be used to conduct the research proposed.

3. FRAMEWORK AND METHODOLOGY

This chapter introduces the methodology that was followed for this study. First is a general outline on how the study was conducted followed by a more detailed breakdown of the techniques that were implemented. Along the way the rationale for the decisions made are also be explained. Following that, will be an explanation of the samples chosen and the hypothesis being tested. The variables for the experiment will be explained and the rationale for why decisions were made about them. Finally, the factors for a successful test will be discussed.

3.1 Overview of the Study

This study assessed how accurately changes in block structure of a mobile device were at classifying a phone and at masquerade detection. Testing was done using multiple machine learning methods to determine if a session belongs to the owner of a smartphone or someone else. It used the 36 subjects from the 2013 Purdue Experiment dataset as the source of its user sessions. The Purdue Experiment was a research project developed by MITRE engineers as part of the PMF project, including the author of this experiment. The Purdue Experiment dataset came from a collaborative research project between MITRE and Purdue University. It was a one of a kind collection of mobile phone images taken while the phones were actively being used by unsuspecting subjects and not created in a laboratory setting.

The unique design of the PMF not only created forensic images of the phones on a regular periodic basis but also allows for reverse tethering. Reverse tethering means the images can be pushed back onto the phones themselves. Allowing any phone to be recreated identically to how it was at any point when a collection was taken. This ability was used in the study to inject attacks into the phones so that the attack's data

would be fully incorporated into the original data. New images were then collected and analyzed to test which algorithms can best detect masquerader sessions. The ability to inject specific attacks into a data collection to study its effects proved novel and advantageous. Being able to determine if changes in device storage can be used for masquerade detection while maintaining user privacy is even more important. This is especially true as big data and mining reveal increasingly more information about device users.

3.2 PMF Explained

Periodic Mobile Forensics (PMF) is a forensic platform for creating and analyzing images from Android smartphones (Guido et al., 2016a). As the dataset used in experiment was from the Purdue Experiment and was created using PMF, it is important to understand how the data was collected, transmitted, transformed, reconstructed, and analyzed. Most of the information in this section comes from already published reports by the MITRE research team that conducted the Purdue Experiment (including this author) that describe PMF and its capabilities.

PMF utilizes differential analysis and several other techniques to create forensic images of smartphones that have been preconfigured with an agent known as TractorBeam (Guido et al., 2016a). TractorBeam then periodically sends data that has been changed back to a secure enclave where it is de-duplicated and stored in a relational database (Guido et al., 2016a). From there the images are reconstructed by the Analysis Framework, which also automates several other forensic processes and detectors. Any artifacts and audit data pulled out by the Analysis Framework are stored in a Mongo database (Guido et al., 2016a). Throughout this entire process there are several preprocessing techniques that occur to guarantee the data is transferred, stored, and reconstructed in a forensically sound and secure manner (Guido et al., 2016a).

The Purdue Experiment was a three month field deployment of PMF run in conjunction with the Computer and Information Technology Department at Purdue University. It was divided into two phases; the collection phase and the analysis phase (Guido et al., 2016a). The collection phase required human subjects for testing and both the MITRE Institutional Review Board (IRB) and the Purdue IRB approved the research before the experiment was conducted. The Purdue Experiment started in September 26, 2013 when 33 smartphones were handed out to human subjects. The smartphones were Samsung Galaxy S3s with unlimited calling plans that included mobile data and wi-fi hot spotting enabled (Guido et al., 2016a). Some minor deception was used to encourage normal phone usage from the subjects. The collection phase ended on January 13, 2014 when the phones were collected and services shut off (Guido et al., 2016a). Post-session consent forms were given to the subjects informing them of the true nature of the experiment. During the experiment, two denied consent to use their data and one dropped out of the experiment early. All three subjects were treated the same and their data was purged before the analysis phase began (Guido et al., 2016a).

The analysis phase started with the reconstruction of images that had been collected and stored from the collection phase. After the data for the three aforementioned users were purged, there were a total of 821 potential images that could be reconstructed (Guido et al., 2016a). The Analysis Framework reassembles the images by inserting the stored bit sequences and offset points into a baseline image as seen in figure 2.6 (Guido et al., 2013). After an image is rebuilt, the Analysis Framework initiates a series of forensic process that extract information from it and stores them in the Mongo database. The audit data stored in the Mongo database was then used by the Predictive Heuristic Interface Layer (PHIL) to develop a profile of standard behavior for the phone. After the profiles for each subject were created, session data was switched around to simulate a masquerading user. Machine learning algorithms were then used to determine which types of data could best be used in detecting the masquerader.

Throughout the PMF process, data originating from the smartphones were hashed, transferred, de-duplicated, and analyzed. The rest of this section breaks down each of the preprocessing techniques and how they were used as well as how forensic soundness was maintained. Section 3.2.1 examines TractorBeam and how it identified changes and passed them to the PostgreSQL server. Section 3.2.2 goes over the PostgresSQL database and the deduplication and storage processes that occurred there. Section 3.2.3 is about the Analysis Framework and the automated processes it uses. Section 3.2.4 is about PHIL and the masquerade detection. Finally, Section 3.2.5 goes over the results and insights that were learned.

3.2.1 TractorBeam

TractorBeam is an agent that is installed on a mobile device. A small modification to the phone's init.rc file starts TractorBeam when the phone boots and provides it the proper privileges to read the block devices (Guido et al., 2016b). Before deploying any phone, TractorBeam ran an initial scan to create an SQLite database containing a SHA256 hash of every 1MB chunk for each block device (Guido et al., 2013). The database is small and stored in an area unavailable to an unprivileged user. At the same time, TractorBeam transmitted every chunk to the PostgreSQL database to create the baseline image. After that TractorBeam was set to run a scan every 6 hours (Guido et al., 2016a).

These scans would compare the SHA256 values of each chunk to the ones stored in the local database. If the hashes did not match the offset for that chunk, it was marked as changed and updated. Every 24 hours, TractorBeam would attempt to establish a secured connection to the RabbitMQ listening service. Once the connection has been made TractorBeam would encrypt and send all the bit sequences from the changed chunks. If the Purdue Experiment had been designed to look at the change in the block structure itself from the start, shorter sessions and different logging could have been used to make the experiment proposed here more effective.

There were several features built into TractorBeam to prevent data loss if the connection was lost or interrupted. One of the most important features was using an asynchronous message queuing system that could rewind and recollect data from interrupted connections. In the worst-case scenario, if the connection was cut for too long, as soon as it could be re-established a new collection entry would be inserted into the database (Guido et al., 2016a).

From this point on TractorBeam continued to run in the background, monitoring for changes in the block structure. The data that was transmitted via RabbitMQ to the PostgreSQL database still had several transitions before it could be analyzed at the secure enclave. The next section covers what happens to the data after it leaves the phone and is transmitted to the relational database.

3.2.2 PostgreSQL

After TractorBeam established a connection to RabbitMQ the data was transferred to a server in the Amazon Web Services Government Cloud running a PostgreSQL database (Guido et al., 2016a). The data was encrypted using public key encryption. To ensure the safety and security of the data, the private key was never saved on the server.

Once the data (changed bit sequences) was transferred to the PostgreSQL database it was de-duplicated before being stored. To de-duplicate the data the SHA256 hash values were compared to the hash values already stored in the database. If the hash had a match, a reference pointer to the matching value was recorded instead of the entire bit sequence. This resulted in a significant storage savings. By the end of the Purdue Experiment the Analysis Framework could rebuild 1,060 forensic images, each 15,025MB (approximately 16GB). Storing each image in its entirety would require almost 17TB of storage space, but PMF's PostgreSQL database was only 325GB in size. The single instancing strategy resulted in a 52.8 times reduction in the amount of required storage space (Guido et al., 2016a).

Data was only stored in the PostgreSQL database for a short period of time. The final resting place for the data was in a secure enclave within the MITRE enterprise network (Guido et al., 2016a). Once data had been transferred to the enclave and successfully decrypted, it was deleted from the database so the space could be reused (Guido et al., 2016a). At this point the Analysis Framework would take over and rebuild the images as needed as well as run the forensic processes to extract information from the images.

3.2.3 Analysis Framework

The Analysis Framework was used to rebuild images as they were needed. It did this by inserting the collected data into the appropriate offsets in the base image, as shown in Figure 2.6. All 821 separate collections from the 30 consenting volunteers and test phones could be successfully rebuilt (Guido et al., 2016a). Once the raw image was rebuilt a series of forensic processes were immediately executed to extract information.

Prior to the Purdue experiment, a series of tests were conducted comparing the images produced by PMF to ones created by Cellebrite's UFED4PC and MSAB's XRY on a battery of test phones. These two products were chosen because they are industry leaders in the realm of mobile forensics. They operate by taking a full physical image of a phone each time it ran as compared to the differential imaging of PMF. The hash values for all three images were a match, meaning that bit for bit every single byte of the images was identical.

The detectors and loggers the Analysis Framework launched were designed to be modular and each gathered specific types of information. Loggers recorded events that occurred on the phone such as changes to Modified, Accessed, Created, and Entry modified times (MACE) (Guido et al., 2013).

Detectors were designed to look for evidence of malicious activity. Some of the detectors, such as detect_droidwatch.py, recorded multiple events while others would

only look for a specific event or file (Guido et al., 2016a). Table 3.1 below is a list of the detectors, events, and event counts that was run against the rebuilt images.

Table 3.1.: Detectors Used in the Purdue Experiment (Guido et al., 2016a)

Detector	Event	Count
detect_app_usage.py	Daily app usage	18964
detect_browser.py	Browser URL	2240
detect_cameraevents.py	Picture taken	1786
detect_cameraevents.py	GPS coordinates added to picture	0
detect_collection_runs.py	Finished collection	821
detect_collection_runs.py	Unfinished collection	128
detect_droidwatch.py	Network status event	73111
detect_droidwatch.py	Process list event	31998
detect_droidwatch.py	Screen event	168232
detect_droidwatch.py	User present event	36446
detect_droidwatch.py	App Event	3905
detect_droidwatch.py	Ringer mode event	8084
detect_droidwatch.py	Shutdown event	442
detect_droidwatch.py	USB event	5473
detect_droidwatch.py	Location provider event	338
detect_droidwatch.py	Contact event	566
detect_droidwatch.py	Bluetooth event	32
detect_camcorderevents.py	Camera recording	96
detect_chrome.py	Chrome usage	167990
detect_firefox.py	Firefox usage	2097
detect_new_calls.py	Phone usage	6744
detect_new_sms.py	SMS/MMS message	12192

continued on next page

Table 3.1.: *continued*

Detector	Event	Count
detect_screenshots.py	Screenshot taken	100
detect_settings.py	Settings changed	20306
detect_map_activity.py	Map searches	17
detect_groupon.py	Groupon GPS recording	115
detect_emails.py	Email message	12427
detect_inspect_apk.py	App installed, app carved	1651
detect_google.py	Google picture	2855
detect_google.py	Google search	1955
detect_gps.py	GPS coordinate capture	300
detect_gps.py	GPS current location	59
detect_gps.py	GPS location history	121
log_file_mace.py*	logs any added or changed files	
log_fls_parse.py*	logs and deleted files	

* These detectors were only partially run on rebuilt partitions and their usage during the analysis was limited

The numbers above do not equate directly to being useful for masquerade detection by themselves. An example of this is detect_emails, which recorded the number of emails sent and received. Having a high or low number of emails is not a indicator for masquerade detection. Oppositely is detect_bootdelta, which indicate malicious use if it is ever above 0. For the purpose of the Purdue Experiment it was chosen to limit events to those characterized as a communication action such as; phone calls, SMS, MMS, and emails (Guido et al., 2016a). These events were chosen because it was believed they would be the most indicative of user behavior and could therefore be used in masquerade detection.

The counts of the audit data were then aligned into groupings called sessions. Each session represented a period of device activity such as hourly or daily usage. Using heatmaps to look at these sessions it was clear that they could distinguish different subjects. Figures 3.1 through 3.3 show examples of hourly session heatmaps from three different subjects over the course of a week. These users had roughly the same number of sessions so that the maps would look the same if their behavior was not different.

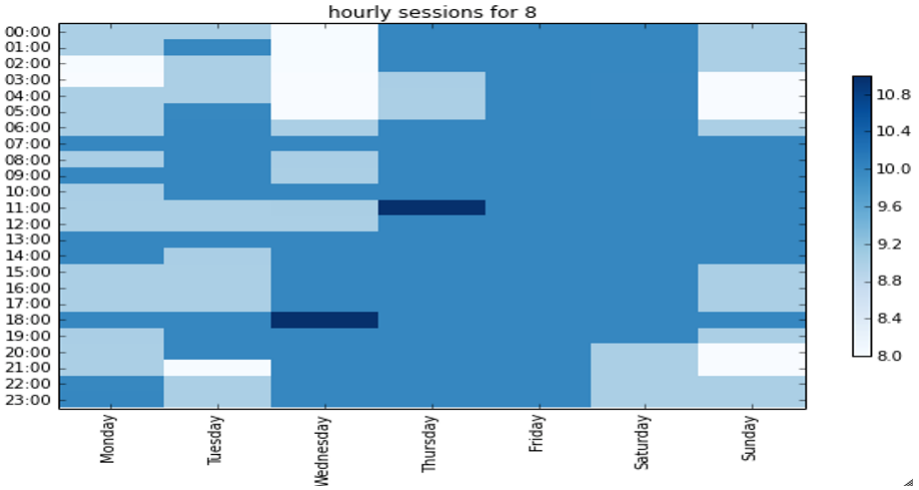

Fig. 3.1. Session heatmaps from phone 8 (Guido et al., 2016a)

Other usage patterns were also developed using 23 separate statistical features. While useful for showing that each user did have distinguishable behavior, it was not enough to tell if someone was masquerading as the subject. To do that the Predictive Heuristic Interface Layer (PHIL) was needed. PHIL was the framework used to test various machine learning algorithms and create user profiles that could be compared to test sessions to determine if that session belonged to the user or a masquerader.

3.2.4 Predictive Heuristic Layer (PHIL)

PHIL itself is a framework built in IPython using the NumPy, SciPy, scikit-learn, and Pandas (Guido et al., 2016a). IPython is an interactive Python shell that provides

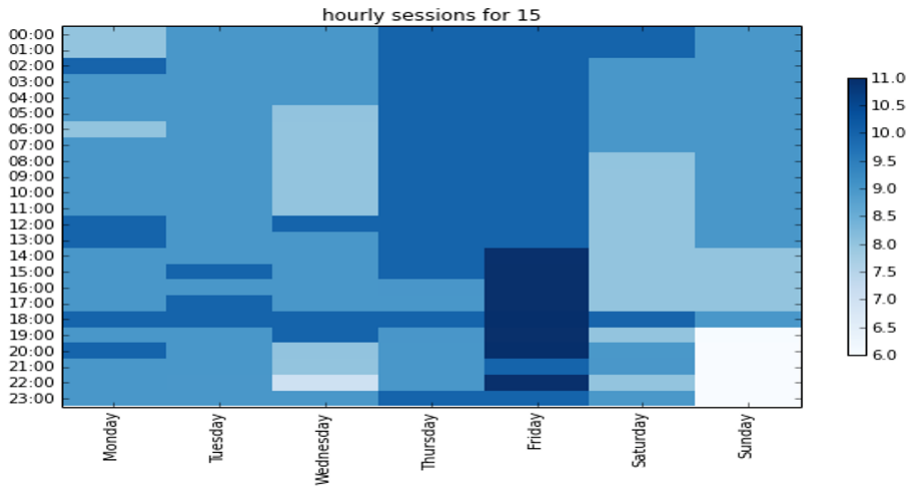

Fig. 3.2. Session heatmaps from phone 15 (Guido et al., 2016a)

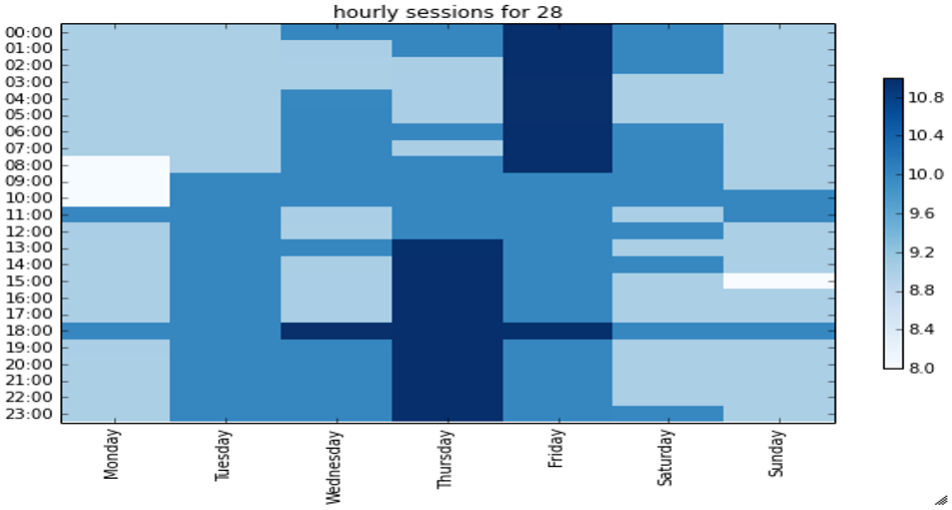

Fig. 3.3. Session heatmaps from phone 28 (Guido et al., 2016a)

some GUI support for the user. The other libraries are scientific and machine learning libraries with many built in features capable of analyzing and visualizing the audit data. The classifiers chosen were Latent Dirichlet Allocation (LDA), Decision Tree, Gaussian Naïve Bayes, Linear Support Vector Machine (SVM), Radial Basis Function kernel SVM, Adaptive Boosting (AdaBoost), Random Forest, and K Nearest Neighbor (Guido et al., 2016a).

An automated tool called a notebook was created for each test set, a combination of a classifier, a subset of statistical features, and session data. Runners were created to incorporate several notebooks so that each test set would not have to be run individually. A leaderboard was established to track which features and classifiers were doing the best at detecting masqueraders (Guido et al., 2016a).

To evaluate the classifiers, a pairing scheme was used that paired two phones at random from the 30 phones in the study. It used K-fold cross validation where K = 6. K-fold cross validation means that a sample is divided into k subsamples where k - 1 samples are used as training data with the remaining sample used as the test sample. Cross validation is repeated k times (the folds) until every sample has been used as the validation test. The results are then averaged to get a single estimated value. This meant that our tests used five sessions training and the sixth was used as the validation test.

For the K-fold cross validation to work, each phone had to have sessions present for every fold and needed a minimal subset of features present in those sessions. If a session did not meet the features requirement, then those sessions were discarded from the paired phones session list. In total, 870 pairings were made after phones that did not have enough valid sessions were discarded (Guido, et al., 2016). Each pair was plotted on a ROC curve using a one algorithm and feature subset, an example of which is seen in figure 3.2. The more area under the curve (AUC) the better the algorithm was at determining if a session belonged to the user or someone else. This is used to maximize the number of true positives while minimizing the number of false positives.

Most of the tests could achieve a perfect result on some of the pairings, but on average were slightly lower. Figure 3.3 shows a box plot that graphs how the various classifiers performed. The best performer was the Random Forest model with a median score of 84% across all pairs. For masquerade detection, this is a very high degree of accuracy. It is possible that with a less random pairing method that this performance could be improved.

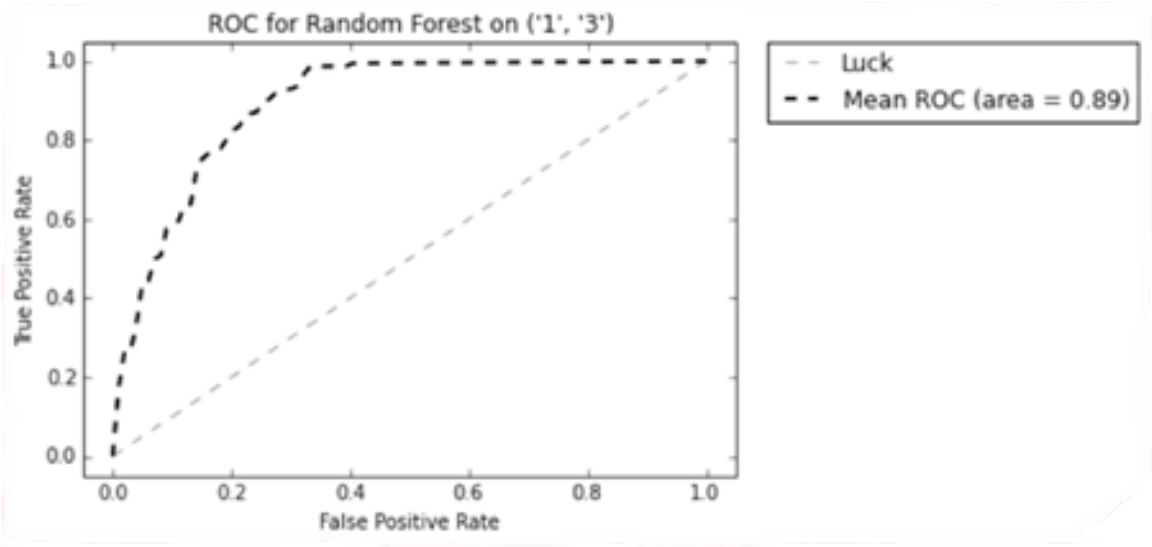

Fig. 3.4. ROC curve showing the comparison of phones 1 and 3 (Guido et al., 2016a)

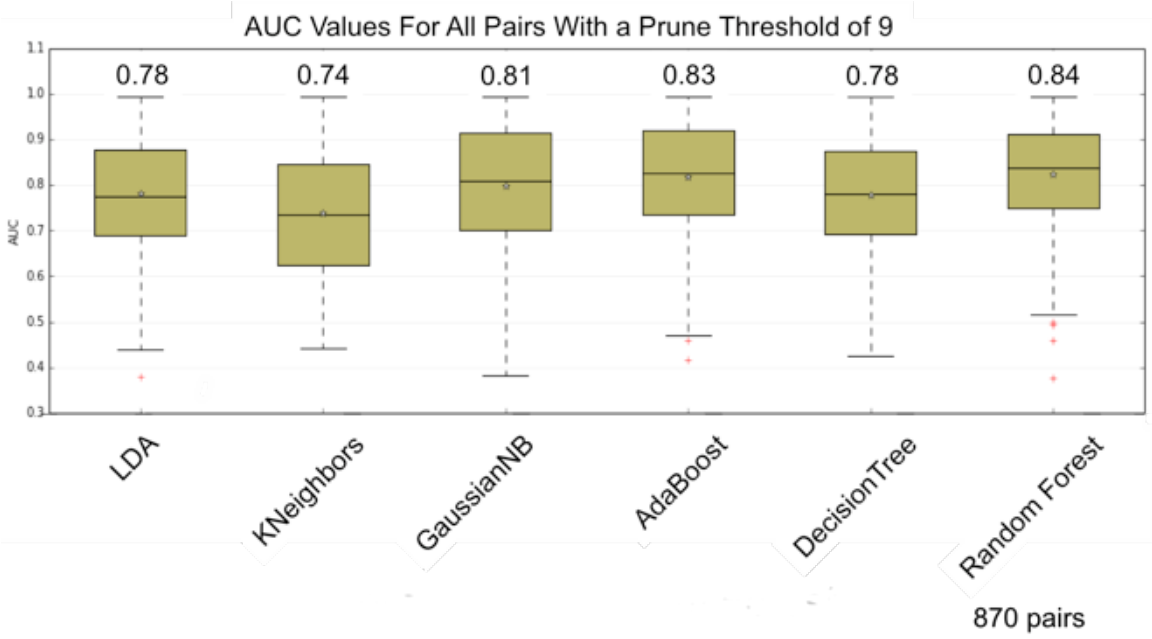

Fig. 3.5. Box plot of classifiers across all pairings (Guido et al., 2016a)

3.2.5 Reverse Tethering

Determining the effectiveness at PMF in detecting masqueraders was the purpose for the Purdue Experiment, but was not the end for testing and improving upon PMF. There were many lessons learned and techniques that were either improved on or developed because of the success of the Purdue Experiment including the proposed experiment of this paper. One of the new features that was developed for PMF is the ability to push a previously recorded session back onto the phone it came from, known as reverse tethering.

Reverse tethering essentially scans the phone and compares the hash values of each block to the hash values of a previous collection. When a difference is discovered, the data from the stored collection is written onto the phone at the same offset. Effectively returning the phone to the exact same state it was in when the collection was taken, bit for bit.

This research took advantage of reverse tethering by restoring a phone to its prior state and injecting an attack into it. After the attack has been conducted, PMF can be used to create a new collection. The new image had the original block structure from the good session except for where the attack has changed it, creating a perfect masquerading session.

Before these masquerade sessions can be created, the specific attacks that were used needed to be determined. The next section describes how to create an attack pool for masquerade detection and security research on smartphones. It also explains which attacks were be used in this experiment and why.

3.3 Attack Pool

To create a pool of attacks that accurately reflects the environment a smartphone will be exposed to, it is important to understand what types of threats the phone will face. Closely related to understanding the types of attacks is understanding the various motivations, tactics, and techniques of an adversary. Knowing these helped

in determining what attacks to expect or what categories would be exploited by the adversary. It is also important to consider what the security mechanisms are detecting and how the attacks will trigger that detection. Without considering it would be possible for the tool to miss artifacts that would otherwise signal an attack occurred. Another important item to understand is what attacks are happening around the globe. There are simply too many ways to compromise a smartphone and too many threats to test them all.

NIST published a website, Mobile Threat Catalogue, to describe and identify the threats posed to smartphones and other mobile devices. They created eleven different categories of threats (NIST, 2016):

- Application
- Authentication
- Cellular
- Ecosystem
- Emergency Mobility Management (EMM)
- GPS
- Local Area Networks and Personal Area Networks (LAN & PAN)
- Payment
- Physical Access
- Stack
- Supply Chain

Each of these categories have multiple threats that fall under their purview with several of the threats having overlap or being able to contribute to one another. The Application category is split into two subcategories, vulnerable applications and

privacy-invasive application. Vulnerable applications refer to any vulnerability that resides within an application running atop the mobile OS. Privacy-invasive application on the other hand consists of malware that gathers and transmits data about the user. Authentication mechanisms has three subcategories, breaking down who/what is authenticating and to what.

Examples of these threats would include lock screen bypasses and applications that capture credentials. Cellular includes any number of threats that can be exploited over-the-air (OTA) via several means. The ecosystem category includes threats that are created when the owner of a smartphone uses added services, such as data back-ups, offered by the OS vendor or device manufacturer. Even though it is supposed to protect both the user and enterprise, the EMM can be abused to allow illicit monitoring of the phone, installation of malware, or theft of credentials. The GPS category covers any jamming or spoofing that can occur. The LAN and PAN category are threats that come from wi-fi, Bluetooth, Near Field Communication (NFC) or similar network that the phone can be connected to. Any form of mobile payment technology can be abused to steal money, or the credentials needed. One of the most dangerous threats to phones is loss of physical control. The stack category refers to the mobile device technology stack which includes any software or hardware required to make the smartphone function. Finally, the supply chain category includes any threats that attempt to compromise a phone while it is being manufactured (NIST, 2016).

These categories are purposely left broad so that all forms of threats can be accounted for. They do not discuss which attacks are the most frequent or even the most likely to occur. Many of these attacks do not have known or published statistics regarding frequency and gathering information on them is difficult. The information most commonly published is about mobile malware, in the application category, and often comes from security companies that have a vested financial interest in promoting the knowledge and fear of mobile malware. This does not make their reports inaccurate but is a potential source of bias in the knowledge base.

Knowing that a vulnerability could exist is not enough for security testing, as actual exploits and attacks need to be used. Instead, examples that represent the types of threats that are occurring should be used. A great deal of research has been done on mobile malware as it is one of the most common threats to smartphones (CyberEdge, 2015; Bach, 2015; Garnaeva et al., 2014; Huang & Stamp, 2011; Miettinen & Halonen, 2006; Shabtai et al., 2010; Shi et al., 2010; Symantec, 2016; Vanja, 2014; Yazji et al., 2014). For example, not all malware is equally prolific and different countries face different distributions of them. The distribution and amounts of malware need to be taken into consideration when developing a pool of attacks for research.

Finally understanding who is using the phone, where it is being used, and which adversary wants what data can provide significant insights into which attacks the phone may face. This use case can be designed to be as specific or generalized as desired. The use case being studied for this research is a lunchtime attack. The term was phrased from computer security because the attacks would occur while the user is away from their desk or at lunch (Tech-Faq, 2015). Smartphones are particularly susceptible to lunchtime attacks because a moment of inattention can leave the phone vulnerable whenever the user sets it down. During this time the adversary may attempt to do anything from accessing files, exfiltrating data, or even installing malware or other apps without the user noticing. Unlike malware infections, there are no statistics for how frequently this style of attack occurs, but they do occur and should be included in the attack pool.

The following subsections break down each of these elements to provide guidance into what attacks need to be included in the pool. Correctly determining this pool will simulate the actual environment and provide more realistic testing results when developing new security tools. Considering these elements may also reveal weaknesses in the tool that would otherwise be overlooked in the testing environment. The following subsection starts this process by examining how attacks are conducted using the Cyber Kill Chain.

3.3.1 Cyber Kill Chain

The seven stage Cyber Kill Chain from Lockheed Martin was designed to model how cyber-attacks are carried out against traditional computer networks. It provides a high level understanding of how network attacks are conducted. With a little modification, it can be applied to attacks against smartphones. The seven stages of the Cyber Kill Chain include (Martin, 2016):

- Reconnaissance
- Weaponization
- Delivery
- Exploitation
- Installation
- Command and Control (C2)
- Actions on Objectives

The first two of these stages, reconnaissance and weaponization, are done before the adversary ever attempts to conduct the attack. Reconnaissance traditionally means identifying targets via the internet. Often this is done by searches or network scans (Sager, 2014). For smartphones, this includes learning about the hardware and the OS involved. Every variant of a phone and OS will require slightly different tools and techniques to exploit it. Once the exploits and tools have been chosen, the adversary can then package them for delivery in a process known as weaponization. These two stages are hard to detect as they have little direct interaction with the target.

Delivery occurs when the weapon is transmitted to the target. This can be done via the internet or in the case of a lunchtime attack, in person. From here the security exploit is ran to give the adversary control. Once control has been established any

tools or software the adversary needs are installed. For persistent threats, C2 is established, allowing the adversary remote control of the phone and its processes. After C2 has been established the adversary then can exfiltrate the desired data off the phone.

Choosing attacks that impact the phone at specific points in the kill chain can be used to determine how early the tool is capable of detecting the attack. If the tool works at an early enough stage in the attack, a response can be done to mitigate the effectiveness of that attack.

While the Cyber Kill Chain describes the general stages of an attack, to understand specifically what is happening during those stages a more in depth model is needed. The next section covers the ATT&CK framework and provides a deeper understanding of the tactics and techniques being used at those stages. Understanding the tactics and techniques being used allows one to choose attacks for the pool that highlight the techniques wanting to be studied.

3.3.2 ATT&CK

The MITRE ATT&CK model expands on the last three stages of the Cyber Kill Chain, installation, C2, actions on objectives, to provide a better understanding of how attacks are accomplished (MITRE, 2015). ATT&CK stands for Adversarial Tactics, Techniques, and Common Knowledge. It consists of nine categories of tactics each listing the various techniques that could be used to affect that tactic (MITRE, 2015). Figure 3.4 shows the nine tactics categories and a few of the various techniques that could be used to achieve them. Like the Cyber Kill Chain, ATT&CK was developed for network security, but can be modified for smartphones and other mobile devices.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
10 items	31 items	56 items	28 items	59 items	20 items	19 items	17 items	13 items	9 items	21 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Command-Line Interface	AppCert DLLs	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Control Panel Items	AppInit DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	File and Directory Discovery	Exploitation of Remote Services	Data from Information Repositories	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Application Shimming	Clear Command History	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Execution through API	Authentication Package	Bypass User Account Control	CMSTP	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Spearphishing via Service	Execution through Module Load	BITS Jobs	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Password Policy Discovery	Pass the Ticket	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Dylib Hijacking	Component Firmware	Forced Authentication	Peripheral Device Discovery	Remote Desktop Protocol	Data Staged	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Graphical User Interface	Browser Extensions	Exploitation for Privilege Escalation	Component Object Model Hijacking	Hooking	Permission Groups Discovery	Remote File Copy	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	InstallUtil	Change Default File Association	Extra Window Memory Injection	Control Panel Items	Input Capture	Process Discovery	Remote Services	Input Capture		Multi-hop Proxy
	Launchctl	Component Firmware	File System Permissions Weakness	DCShadow	Input Prompt	Query Registry	Replication Through Removable Media	Man in the Browser		Multi-Stage Channels
	Local Job Scheduling	Component Object Model Hijacking	Hooking	Deobfuscate/Decode Files or Information	Kerberoasting	Remote System Discovery	Shared Webroot	Screen Capture		Multiband Communication
	LSASS Driver	Create Account	Image File Execution Options Injection	Disabling Security Tools	Keychain	Security Software Discovery	SSH Hijacking	Video Capture		Multilayer Encryption
	Mshta	DLL Search Order Hijacking	Launch Daemon	DLL Search Order Hijacking	LLMNR/NBT-NS Poisoning	System Information Discovery	Taint Shared Content			Port Knocking
	PowerShell	Dylib Hijacking	New Service	DLL Side-Loading	Network Sniffing	System Network Configuration Discovery	Third-party Software			Remote Access Tools
	Regsvcs/Regasm	External Remote Services	Path Interception	Exploitation for Defense Evasion	Password Filter DLL	System Network Connections Discovery	Windows Admin Shares			Remote File Copy
	Regsvr32	File System Permissions Weakness	Plist Modification	Extra Window Memory Injection	Private Keys	System Owner/User Discovery	Windows Remote Management			Standard Application Layer Protocol
	Rundll32	Hidden Files and Directories	Port Monitors	File Deletion	Replication Through Removable Media	System Service Discovery				Standard Cryptographic Protocol
	Scheduled Task	Hooking	Process Injection	File System Logical Offsets	Securityd Memory					Standard Non-Application Layer Protocol
	Scripting	Hypervisor	Scheduled Task	Gatekeeper Bypass	Two-Factor Authentication Interception					Uncommonly Used Port
	Service Execution	Image File Execution Options Injection	Service Registry Permissions Weakness	Hidden Files and Directories						Web Service
	Signed Binary Proxy Execution	Kernel Modules and Extensions	Setuid and Setgid	Hidden Users						
	Signed Script Proxy Execution	Launch Agent		Hidden Window						
	Source			HISTCONTROL						
	Space after Filename			Image File Execution Options Injection						

Fig. 3.6. MITRE's ATT&CK Framework (MITRE, 2018)

Any attack against smartphones could require several different tactics and multiple techniques to be successful. Knowing what tactics and techniques will be used lets one make an informed decision on what types of security mechanisms will be required to detect them. For instance, privilege escalation is often done using a security exploit to gain root access (Lohrum, 2014). This could be done by swapping out SD cards with one that is preloaded with an exploit such as towelroot. Once the phone is rooted, the adversary then has full control of the mobile phone. They could potentially embed software wherever they like, including editing the system/recovery partitions or even installing a new OS. Rooting the phone also can provide persistence, another tactic category that could be highly used.

Malware often requires a means of establishing persistence to be effective. This is because malware can be used to enable several other tactics and once it is installed on the phone it is unlikely it will ever be noticed. Only 14% of users have antivirus running on their phones and the antivirus that exists is often woefully behind the times (Ruggiero & Foote, 2011; Tapellini, 2014). That means many users are not looking for malware on their phones and any attempts to hide it are likely to work. As far back as 2012 the Ginmaster malware started using the obfuscation of class names, encrypting its C2 code, and using polymorphism techniques previously only found in traditional computer malware (Svajcer, 2014). While the clear majority of users are not checking for malware on their phones, the malware writers are already prepared to use defensive evasion tactics to ensure their malware remains unnoticed and effective.

Smartphones are a potential smorgasbord of sensitive data including email, calendars, contact information, passwords and point of sales (Ruggiero & Foote, 2011). Often a malware author wants access to banking accounts and credentials, common goals for smartphone malware (Lookout, 2014). Kaspersky Lab reported that there were nine times as many banking Trojans in 2014 as there were in 2013 (Garnaeva et al., 2014). Much of this data is stored in SQLite databases and sometimes in plain text. Any of this data could be targeted for exfiltration.

Exfiltration includes any techniques that aid an adversary in removing information from the smartphone (MITRE, 2015). This can be done in several different ways. One common technique is using the available networks (Cellular, Bluetooth, or WiFi) to transmit data to another location. This is often handled by a C2 system and can use the same channel as the C2 or an alternate one (MITRE, 2015). Another technique that can be used against smartphones is using physical mediums to transmit data. An adversary could potentially connect the phone to a computer through a cable and physically copy data. Depending on the amount of time available and the location this may be noticeable. Another physical method would involve switching the SD card and copying data over. Once the transfer is complete the SD card can be replaced with the original and no one would be the wiser.

These were just a few examples of the tactics and techniques that can be employed in an attack. It is important to understand these different tactics and the techniques because they determine what the adversary is doing and how they are going to accomplish it. This information should not only guide how a new security tool is developed but also how it is tested. Now that the life-cycle of any given attack can be understood it is important to also understand what types of attacks are occurring in the wild and to what extent. The next section covers the distribution of attacks both by number of occurrences and geographical location.

3.3.3 Distribution of Attacks

The lack of security, the information, and money accessible to smartphones has proven to be irresistible to cyber criminals. The number of attacks facing mobile phones has been increasing at nearly an exponential rate. A clear majority of attacks on smartphones are malware. In ten years, the amount of malware for mobile phones has increased from a mere handful to over 30 million in 2018 (Samani et al., 2019). Symantec reported blocking over 10 thousand malicious mobile apps per day in 2018

as well (Symantec, 2019). Over time as new vulnerabilities are found and old ones are patched, the landscape of malware continually shifts.

Not all countries or even organizations are equal when it comes to cyber-attacks and threats. All malware does not spread evenly throughout the globe as can be seen in the figures 3.5 through 3.7. The reasons for committing cyber-crime are as myriad as the individuals and organizations that commit them. The culture, customs, and laws of one region can make a particular attack more popular than others. Something as simple as a popular app or phone in one country may cause a certain family of malware to be more effective. At other times politics and economics influence what happens as nation states square off against one another. No matter what the reason, the attack landscape is an ever changing and evolving environment.

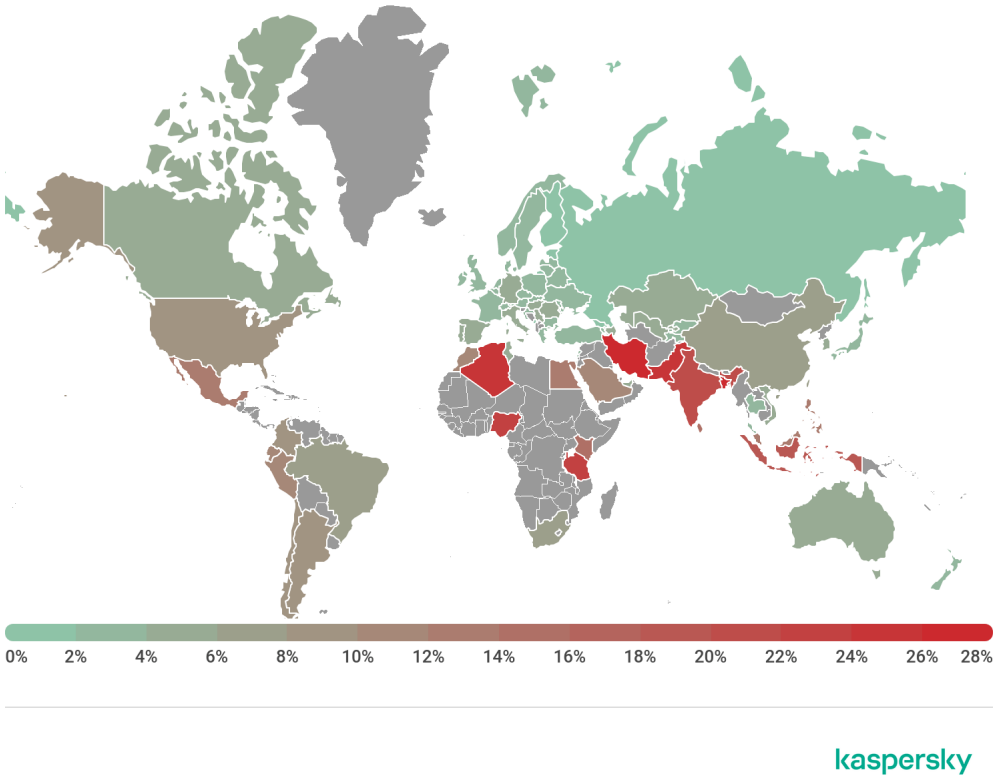

Fig. 3.7. Geographical distribution of mobile malware (Chebyshev et al., 2019)

Table 3.2.
Top 5 countries by share of users attacked by mobile malware (Chebyshev, 2019)

Rank	Country	%
1.	Iran	28.31
2.	Bangladesh	28.10
3.	Algeria	24.77
4.	Pakistan	24.00
5.	Tanzania	23.07

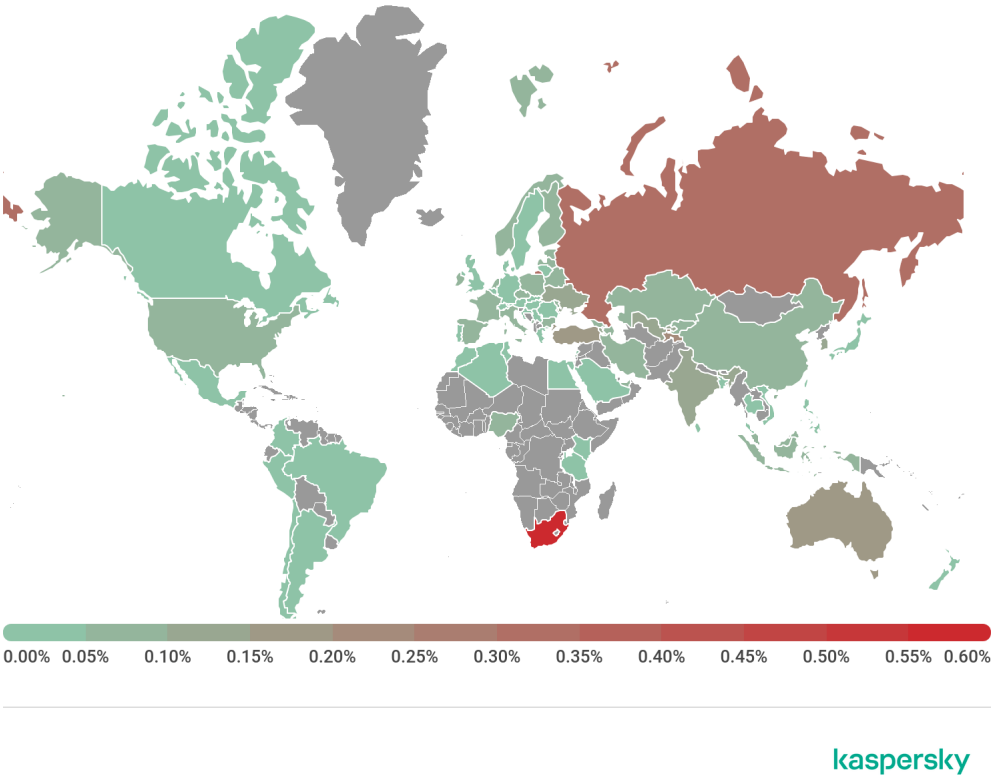

Fig. 3.8. Geographical distribution of banking Trojans (Chebyshev et al., 2019)

Table 3.3.
Top 5 countries by share of users attacked by mobile banking Trojans (Chebyshev, 2019)

Rank	Country	%
1.	South Africa	0.64
2.	Russia	0.31
3.	Tajikistan	0.21
4.	Australia	0.17
5.	Turkey	0.17

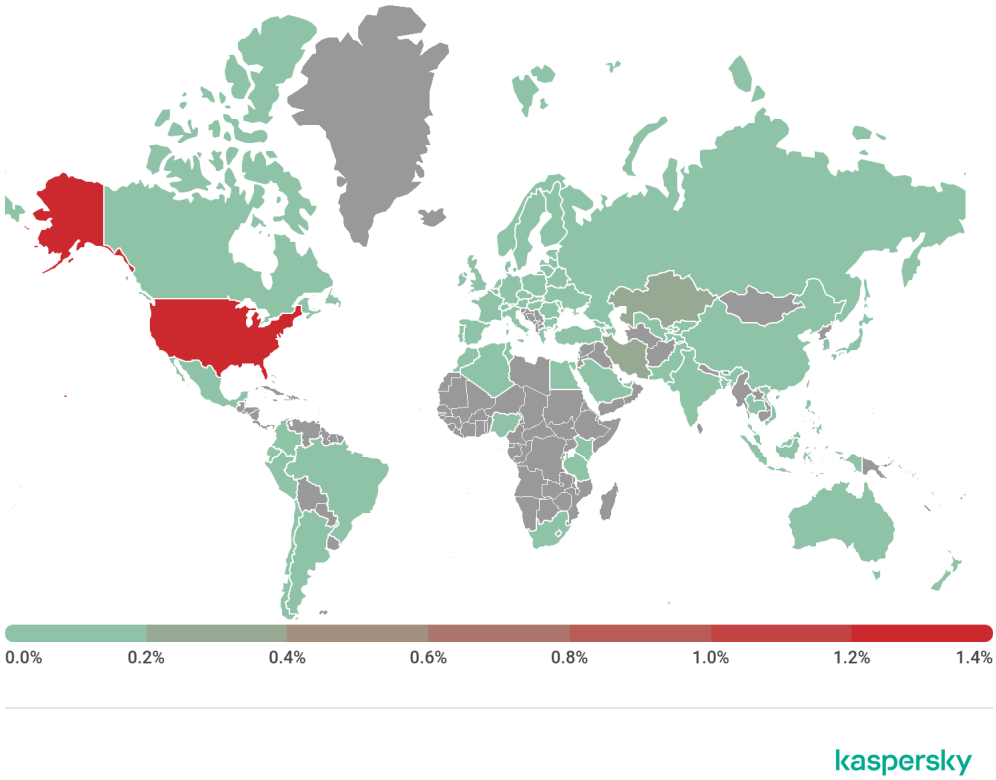

Fig. 3.9. Geographical distribution of mobile ransomware (Chebyshev et al., 2019)

Table 3.4.
Top 5 countries by share of users attacked by ransomware (Chebyshev, 2019)

Rank	Country	%
1.	United States	1.58
2.	Kazakhstan	0.39
3.	Iran	0.27
4.	Pakistan	0.16
5.	Saudi Arabia	0.10

As can be seen in the maps, geographic dispersion of malware is not equal. Not only do the amounts of malware infections vary, the types of infections also change by country. Each country has its own peculiarities that alter the nature of what occurs in them. In 2014 weak cyber-laws and a lack of enforcement have combined with poverty in Brazil made banking malware the most common threat there whereas by 2018 improvements have significantly reduced the amount of banking infections (Team, 2016; Chebyshev et al., 2019). In 2018 McAfee found that the South Africa had the most banking trojan infesctions, while ransomware was more infectious in the United States (Chebyshev et al., 2019). While simplified, these maps show that different countries face different threat environments.

When choosing malware samples, a variety of malware should be used to test a MDS. Special attention should be payed to the most common malware that the MDS will encounter. According to Spreitzenbarth (2016), there are approximately 208 different families of Android malware (Spreitzenbarth, 2016). A sample from all 208 different families would require extensive testing that may be too intensive for both time and resources. Many of these families are also isolated to specific regions of the globe or are completely outdated. To create a sample pool to choose from, the current most prolific malware families should be examined. The absolute positions of these families may change based on the company detecting them and the regions they

are monitoring. Table 3.5 is a list of the top 10 most infectious malware as reported by the security firm Symantec for 2018.

Table 3.5.
Top 10 most infectious mobile malware of 2018 (Symantec, 2019)

Malware Name	Percent
Malapp	29.7
Fakeapp	9.1
MalDownloader	8.9
FakeInst	6.6
Mobilespy	6.3
HiddenAds	4.7
Premiumtext	4.4
MobileSpy	2.8
HiddenApp	2.5
Opfake	2.0

Simulating a realistic attack environment would require attacks to occur in similar percentages. While using top 10 lists will provide a generic base for the amount and type of malware that should be used in a simulated environment, each country will have its own differences. These differences should be accounted for when designing the simulated environment.

The ARGUS Cyber Security Lab has created the Android Malware Dataset (AMD) that consists of multiple samples of the major Android malware families. They also created a detailed table, the Android Malware Behavior Table, that allows the user to: quickly find out what type of malware a sample is, what it does, how it installed, what privileges it uses, how it many variants it has, how many samples were collected, and several more items that could be of interest to a researcher (ARGUS, 2019). It also provides links with detailed data and virus reports the malware families and the

variants. The AMD project provides a means to select the types of malware and the various features that the researcher wishes to examine. ARGUS also provides a copy of their dataset to universities and research labs. These are known good samples of malware that are suited for conducting research. Having safe samples to work with is also an important factor when determining which malware should be chosen to work with. As such, there will be a restriction on this research that limits it to samples that can be found within the AMD.

Samples from the most infectious families and the current most prevalent malware will better simulate the overall threatscape that a smartphone is facing. It also will reduce the number of needed tests to a more reasonable amount. While malware is the most common threat being encountered in the wild it is by no means the only threat to a smartphone as the NIST catalogue proves. One should consider where a phone is being used and by whom in order to include a more representative view of the threats being faced. The next section covers the idea of a use case and other threats that may need representation in the attack pool.

3.3.4 Use Case and Scenario

Understanding of who is going to be using the phone and where it will be used can also help determine what attacks should be included in the attack pool. A use case simply describes who owns the phone, where it is located, who might be interested in it, and why. A smartphone owned by a child in the Midwest of the United States faces a very different scenario than a phone owned by business person in New York. A good MDS should be able to identify when anyone other than the owner is using the phone for any reason.

Outright theft of the phone, while an option does alert the owner that something has happened. It may take a little time, but a responsible person will eventually note the theft occurred and have it reported. The may even attempt to remotely lock, wipe, or locate the phone. This gives the adversary some time to look at the phone,

but that time is limited. If the adversary's goal is to fence the phone for profit, then these issues might not matter. If the adversary is more insidious, they may try attacks such as accessing the phone and transferring data off it or installing targeted malware. This is known as a lunchtime attack and it leaves the owner oblivious to what has happened while providing the adversary persistent access to data on the phone and potentially access to any network it connects to (Tech-Faq, 2015). These types of attacks typically belong to the physical access category of the NIST catalogue.

As discussed in section 2.1, many users still don't employ even a simple PIN lock to keep strangers from using their phone (Anderson, 2017). To make matters worse, many exploits exist that can easily bypass screen locks,other security mechanisms, and grant root level privilege. With just a few minutes of inattention, an adversary could easily swap a phone's SD card with one preloaded with a security exploit such as towelroot and gain root access, complete control over the phone. With root level privilege the adversary may install or modify any software on any partition including the recovery and system partitions. This would give them a persistent foothold that will be nearly impossible to detect and difficult to remove.

Even if the adversary chooses to not install any malware, they could simply start exfiltrating data directly from the phone. A quick SD card swap and then they could copy data directly over and then replace the original, once again leaving the owner none the wiser. The adversary could also start forwarding sensitive emails and other data to themselves. When was the last time you looked in your sent items to see if an email you did not send was in there? Sometimes, simply knowing who is in your contact list and how to reach them may be all that is desired. The point is, if you turn your back on your phone for even a brief period it may come under attack in ways you never thought about and will likely never detect without a well-tested security tool.

Even if one does not turn their back on their phone, it may come under attack without one's knowledge. Smartphones are highly networked devices with not only 4G cellular connections but wi-fi and Bluetooth antennas as well. Many people who

use Bluetooth leave the service on and unprotected even when they are not actively using a Bluetooth device. Attacks such as Bluesnarfing and Bluebugging can give an adversary remote access to the phone's contact list, emails, text messages, and even allow them to listen in or place phone calls (HubPages, 2016). These attacks fall under the NIST LAN & PAN category. They have limited range but don't require the adversary to physically access the phone.

Another series of attacks, in the cellular category, doesn't require the owner to enable anything on their phone as it takes advantage of the Open Mobile Alliance – Device Management (OMA-DM) network. In their 2014 presentation, Solnik and Blanchou demonstrated that they were able to exploit the OMA-DM tool embedded in most smartphones by the manufacturer or carrier (Solnik & Blanchard, 2014). With control over this tool they could send OTA commands with root level privilege. This allowed them to tell the phone to do whatever they wanted, make calls, transfer data, or even process updates that they initiated.

Some of these attacks may seem a little far-fetched, but they do occur. For example, in November 2016 it was reported that a VP at a global technology company had spyware installed on his smartphone when an adversary gained physical access to the it (Zorz, 2016). Depending on who you are and what you are worried about, these attacks can be very important to defend against. Therefore, understanding the use case for the phone is important when designing security tools. If it is not considered and only malware is examined, then highly damaging attacks could slip by the tool when it is deployed in the field. In the high stakes worlds of politics, military might, and even corporate finance, a single successful attack could have dire consequences. The next section of this paper compiles all the information together and discusses how decisions should be made regarding how to create the pool of attacks that should be used when testing new security tools.

3.3.5 Attack Decision Making

This section has examined the life cycle of attacks, the techniques and tactics used in them as well as some common and uncommon attacks. This provides a good basis for making rational decisions about which attacks to include when creating a pool of attacks for testing security tools. The closer the pool is to being representative of what will occur in the wild, the more accurate the results of testing and validation will be.

There are no hard and fast rules regarding how many of and of what type of attacks should be included. Due to the near exponential rise in malware it is obvious that some of the attacks in the pool must be malware, but how many and which? The larger the sample size the more accurate the testing will be but the more expensive testing will be in time and resources. The types of malware and how often they occur can also vary dramatically based on where the phone was located. For this reason, the geographic region chosen to be studied was the United States and the top five malware from that region was included in the pool. This represented about 25% of the malware that was occurring in the wild in that region (Michael, 2016).

Other malware obviously exists, but it occurs so infrequently that they do not necessarily need to be included in the test pool. The 10th ranked malware, COUDW, in F-Secure's list for example only represents .2% of malware infections. The relative percentage that each piece of malware occurs in can also be used to weight how often they are used in the testing environment. Combine this with the frequency that phones face a threat (25% each month) and effective representation of the environment can be created (CyberEdge, 2015). Any malware samples that are new and noteworthy or are specifically desired to be tested should also be included in the pool. For example, when ransomware first started increasing in occurrence it may not have made a top ten list but was important enough to warrant study.

Attacks from the use case for the phone also needs to be included in the attack pool. For the research being conducted the chosen use case was a lunchtime attack

against a smartphone. Determining what should be included from the use case is trickier than determining the malware that should be included because the possibilities are highly varied and there are no statistics for how frequently they occur. The closest research was designed to see how frequently people snoop on another person's phone. Marques, Muslukhov, Guerreiro, Carrico, and Beznosov (2016) conducted a self-reported survey that found 31% of participants had conducted this lunchtime attack sometime in the last year. It does not mention how frequently this activity occurred nor does it include any other attacks.

The last items to be considered while exploring the use case is that of the Cyber Kill Chain and or ATT&CK. If the security tool is supposed to trigger at a specific point in the chain or detect a particular technique, then special attention needs to be paid to make sure attack pool includes examples that meet those requirements. These attacks can be additional malware or attacks derived from use cases.

3.4 The Current Study

The experiment conducted in this paper utilizes MITRE's PMF framework to study if changes in a smartphone's underlying block structure over time can be used to develop a unique user profile. Profiles for each phone were created using five different two-class machine learning algorithms and one one-class algorithm. For the first hypothesis, user behavior will create unique patterns that can be identified, success will be determined by how well the legitimate sessions from a given phone can be classified. The success of the second hypothesis, masqueraders behavior is different that the user's behavior and will leave different patterns in the device, depends on how accurately a legitimate user session can be distinguished from an illegitimate one. For the purposes of this study, an illegitimate user can be a person conducting an attack or undesired software such as malware.

Figure 3.10 shows how the original user images were recorded by PMF for the Purdue Experiment. Figure 3.11 shows how the process is reversed and the original

data is restored using reverse tethering. From there an attack is performed and the changes are injected into the phone exactly as if the attack had occurred in the wild, creating one of the most realistic masquerade sessions ever developed. The original PMF process then repeated and a new collection run is created and stored within the PostGRES database. The machine learning algorithms will then be tested against both the original legitimate user sessions and the newly created masquerader sessions to see how accurately assessing block structure changes of a mobile device can be in detecting masquerading users, mobile malware, and the exploitation of the device.

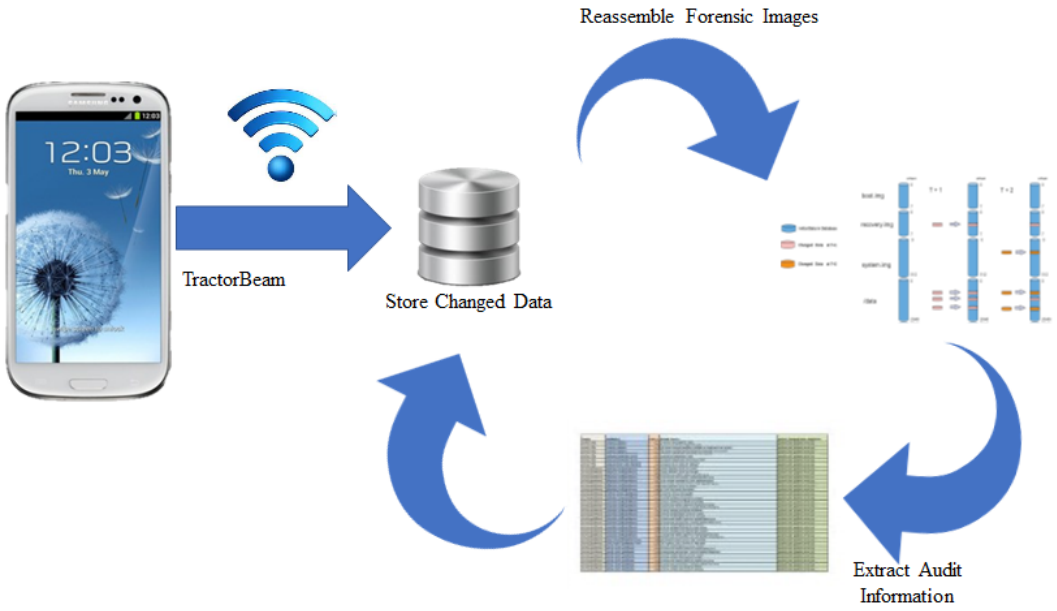

Fig. 3.10. PMF Process (Guido et al., 2016a)

The attack pool for this study was chosen because they are representative of attacks that could have occurred at the time the phones were in use. According to the Symantec (2017) on in thirty-six smartphones experiences an attack each month. That would mean over the course of the Purdue Experiment at least one phone a month should have experienced some sort of attack. Some of the phones could be

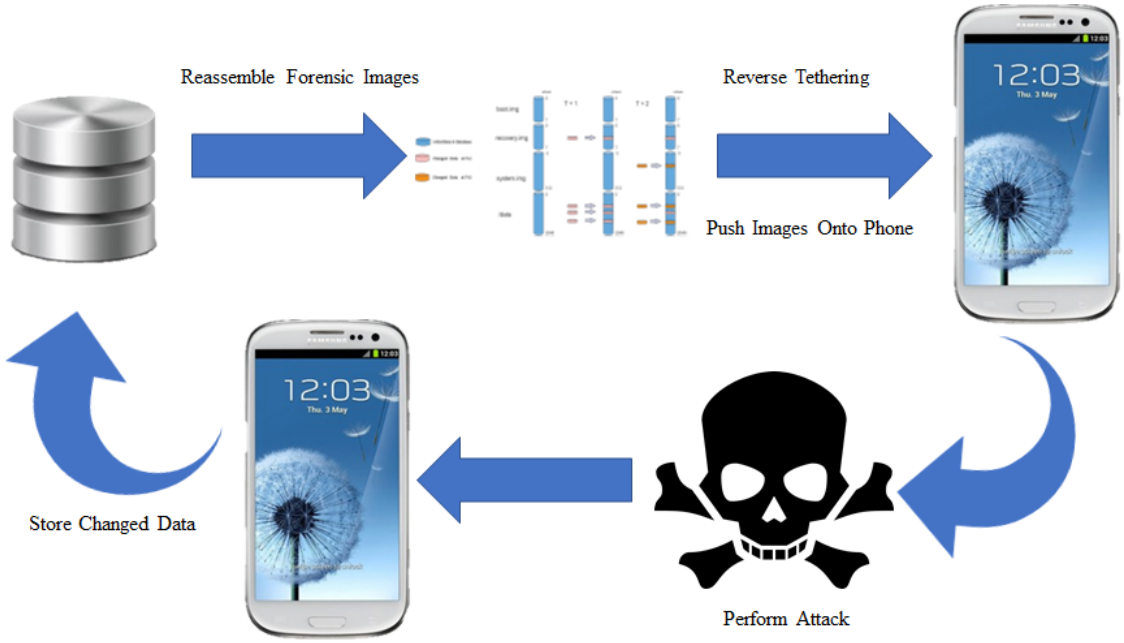

Fig. 3.11. Reverse Tethering Process *parts of image from (Guido et al., 2016a)*

subject to malware, others to snooping, or even third-party intrusions. For this reason, a variety of attacks were chosen.

The first attack for each phone was chosen as a matter of necessity. This was the deletion of the phone's security files. Specifically, the deleted files were:

- Gesture.key
- Locksettings.db
- Locksetting.db-shm
- Locksettings.db-wal

This was done to ensure the researcher could unlock the phone to conduct the additional attacks. The phones were imaged after this to see if this action was detectable. These files are small and do not change frequently, but their removal is indicative of

malicious behavior and represent a large security risk. It would be important for this system to alert if these actions are taken.

From the use case, two different attacks were conducted. The first attack was installing an application and the second was uninstalling an application. As this was a pilot study it was desired to see if this behavior would be considered legitimate or illegitimate. Both actions can be taken by the user, but they can also be taken by a masquerader that takes possession of the phone. For security and forensic purposes, it was desirable to know how these actions would be classified by the algorithms.

For the first attack, the application installed was Plants vs. Zombies, a mobile game. This app was chosen for convenience and because it was able to function on the Samsung Galaxy S3s used in this experiment. The version number was 1.1.44-52 and it was 9.1MB in size. An apk file for the app was loaded onto a micro SD card that was inserted into the phone and then the app was installed from the SD card. Here the objective was to see if the installation of an application can be detected. The chosen app has a below average size for an Android app, but is still large enough to cause changes to the storage space (Boshell, 2017). While this particular app is harmless, a different one may not be, even if it is installed by a legitimate user. Whether the app is harmless or not, it could represent a security risk and the system should alert for further investigation. Knowing if this action is detectable is there for extremely important.

For the second attack, the deleted application was Angry Birds, a mobile game. It was installed on all the phones before they were given to the subjects as part of the standard package of apps. While it is normal for users to occasionally remove apps, it was not known if this action could be detected by the proposed system. As with app installation this action could represent a risk and be desirable to be detected.

The AMD was used to provide the malware and ensure that different types of malware are included such as the banking Trojan and adware that were used. Originally a backdoor agent, ransomware, and spyware were also wanted, but were excluded

from the study do to time constraints. The following malware were chosen for this research:

- BankBot (Banking Trojan)
- Dowgin (Adware)

This malware was chosen because it represents variety of commonly occurring malware from 2013. Any of these samples could have potentially been contracted by the phones while the Purdue Experiment was being run. These types of malware are still common today, but the variants have changed. There are also known valid samples in the AMD to ensure the right malware is being tested. Care was given to make sure that the malware selected will make changes to the hard drive that can be potentially detected by the algorithms.

Malware was installed by transferring it as an apk from an SD card to the phone's data partition and then running it. The phones did not have an active data plan preventing them from connecting to a provider network ensuring user data remained safe. Research was also conducted within a Faraday cage to prevent any signal from escaping and ensuring the malware could not spread or risk subject data from being exposed. A server was set up using INetSim to provide any necessary responses the malware needed to function. INetSim is a simulation suite designed to mimic DNS and other internet services inside a malware examination lab (Hungenberg & Eckert, 2019). The phones connected to the server via a wireless network using a router dedicated for this research. Once the malware was installed on the phone, it was allowed to run, making whatever changes it would normally make. There was no connection to the internet available to the INetSim server, the router, or the phones. Afterwards, the phones were scanned and forensically imaged using PMF. This new image represented a new session that was tested using the machine learning algorithms.

After the attacks were concluded, the phones were wiped, removing all traces of the subject's data and the attacks. These five attacks represent a small portion of the

threats faced by smartphones. The goal was to see if these attacks modify the phone's underlying block structure in a manner that is varied enough from the user's normal behavior for the machine learning algorithms to properly identify which sessions have attacks in them and which ones do not. Testing each of these attacks against the pool of phones will show which attacks are best detected and which algorithms are the most sensitive. To do this, attack sessions need to be generated. The next section covers how these sessions will be created.

3.5 Creating Attack Sessions

One of the most difficult parts of detecting masqueraders on any electronic device is coming up with a valid example of an attack. Many experiments such as (Schonlau et al., 2001; Maxion & Townsend, 2002; Mazhelis & Puuronen, 2007a), and the original PMF masquerade detection experiment train on actual subject data from the training dataset and then insert sessions from other subjects in the testing datasets. The idea is that when the original subject's actions are extracted and examined that the differences between the subjects will become apparent. There are several reasons to do this, one of the most compelling being that session data is set after it is recorded and can't be modified.

The problem with this method is that it does not generate actual attack data. This experiment used primitive data in the sense that it is examining time and locations of data change. Even if users started with physically identical phones, their actions over time would diverge and data would be recorded in different places. Using an alternate subject's data in the testing phase should by default automatically fail as everything would have been recorded in physically different places. The tables below show a simplified hypothetical of why just comparing different user's sessions to each other would fail. Each cell with an app listed is a changed block on the phone. Even if the subjects are using identical apps, time will cause the physical layout of the

phones to drift apart making their sessions look nothing alike as seen in Tables 3.6 and 3.7.

Table 3.6.
Example of Where Data is Stored for Subject A

Facebook			
		Email	
			Instagram
	Chrome	Chrome	
	Twitter		

Table 3.7.
Example of Where Data is Stored for Subject B

Facebook			Chrome
		Twitter	
			Camera
			Camera
Email	Instagram		

This experiment leveraged reverse tethering (section 3.2.5), a unique capability developed for PMF that allows it to inject prior session data back onto a phone and recreate the working state of the phone from any point in time that a previous collection was made. It works by scanning the phone and comparing the hash values of the blocks to the ones stored in the PostGRES database. When a difference is detected it writes the stored data over the scanned block offsets. This forensically duplicates the phone's physical layout at the time the collection was made. From here an attack from the pool was injected, modifying the original data session with the changes that occurred from an actual attack. Table 3.8 shows what the same subject

A's session from before would look like after it has been recreated and an attack run against it.

Table 3.8.
Example of an Attack Session Against Subject A

Facebook			Attack
		Email	Attack
			Instagram/Attack
	Chrome	Chrome	
	Twitter	Attack	

Creating another PMF collection run at this point, allowed the attack data to be recorded in the database. The detection algorithms were then tested against this newly generated dataset. This methodology allowed for the proposed experiment to function by keeping user data in the correct physical location while having actual attack data injected into it. A successful prediction in this experiment would be able to determine that this new session, which came from the phone owned by the subject, had been used by someone else and would thus trigger a potential compromise/alert status. Depending on the necessary follow up, this could trigger automated mitigation responses such as alerting security personnel, disconnecting the phone from the network, or disabling apps and other features of the phone. A full forensic investigation could even be conducted if necessary.

The original subject sessions chosen to be injected with attack data were quasi-randomly selected. Quasi-randomness occurs because some of the phones used in the original experiment no longer functioned and could not be restored to their original states. Of the phones that remained, selection was randomized across all of them and included all the recorded sessions. The forensic image for the chosen session was pushed back onto the phone and an attack conducted. After the attack sessions were generated, the phone was reimaged using the same PMF method that created the

original sessions. It was then up to the machine learning algorithms to determine if the attack sessions could be identified. The next section describes how this occurred.

3.6 Testing and Training

This study tested two types of machine learning methods. The researcher first compared the accuracy of the eight machine learning algorithms running a two-class test. The second one used four different one-class SVMs to determine how well outlier detection can be used to classify a single phone's sessions. The attack sessions from each set of tests should be determined to not belong in the same classification as the actual sessions.

The first portion of this research tested a series of two-class comparisons using eight different machine learning algorithms. The following algorithms were the ones chosen for this experiment:

- Multilayer Perceptron
- Two Support Vector Machine (SVM)
  - Each with k-fold Cross Validation (CV)
- Two Random Forest
- Majority Voting Classifier

They were chosen to test how different algorithms and settings affect the success of the classification. The Random Forest algorithm was chosen specifically because it was the most successful at classifying phones in the original Purdue Experiment and provided a means of comparing the two experiments. The comparisons that were run compared each phone's sessions to a set of sessions created from five other randomly selected phones. Class 1 was the phone being tested while the comparison phones in each test were considered class 0. Each of the experiments was conducted using training/testing splits of 80/20, 70/30, and 30/70. The training data was used to

create the model while the testing data was used as a means of checking the model's validity. The machine learning models generated were then used to test the attack sessions. This tested how similar the attacks were to the actual user session. For these tests, a successful test would result in the attack session being classified as class 0.

The second portion of this experiment used a one-class SVM to compare a phone's sessions to all its other sessions. The idea behind a one-class test is that normal session will be clustered together while abnormal sessions would show up as outliers. This test showed if a phone by itself can be a classified using only its own data. Models were generated by using the same 80/20, 70/30, and 30/70 training/testing splits as the two-class experiments. After the models were created, the attack sessions were tested. A successful prediction would have resulted in the attack sessions being determined to be outliers. The next sections go into more detail on how the models were generated.

3.6.1 Two-Class Machine Learning Method

This being a pilot study, several algorithms were used to find out which one performed the best. Two SVMs were created with slightly different settings. The first SVM, referred to as SVM1, used a C parameter of 1 and the second, SVM2, of 20. "The C parameter trades accuracy for speed. Lower values of C make the decision function simpler (faster), at the cost of accuracy (Learn, 2019d)." Here the decision was made to see how accuracy would change under vastly different C parameters.

Separate models of both SVMs were also made using the same parameters, but applied k-fold Cross Validation (CV) to them as well. "K-fold CV splits the training set into smaller sets called folds. The model is then trained on all but one of the folds, using the last one to validate the model (Learn, 2019d)." This is repeated until every sample has been used as the validation sample. While computationally expensive, this method can be advantageous on small datasets such as the one being used in this

experiment. The first SVM was given a CV value of two and the second CV a value of ten. Again, this was done to better understand the effects of the machine learning settings on the accuracy of the model.

The next set of classifiers were RF classifiers. RF classifiers are ensemble classifiers that work by creating multiple decision trees. Each tree makes a prediction and the classification with the most predictions is chosen (Yui, 2019). Generally speaking, the more trees that are in the model the more accurate it will be at the cost of increased computational difficulty. As with the SVMs, each RF classifier's settings were slightly different to study the effects on the models' predictions. For the RF classifiers, the n_estimators parameters was changed. One was set to 10 and the other to 50. The n_estimators parameter controls how many trees are in the forest, the default being 10. For this research it was desired to know if the increase in trees would significantly improve the accuracy of predictions.

A multilayer perceptron (MLP) was also created using a Limited-memory Broyden-Fletcher-Goldfarb-Shanno (lbfgs) solver as it can perform better on smaller datasets (Learn, 2019b). MLPs work by taking an input layer and an arbitrary number of hidden layers that then proceed to an output layer (Skymind, 2019). The hidden_layer_size was reduced from the default of 100 to 30 because of the small dataset used, where the average number of sessions for each phone was 30.

The final classifier created was a Voting Classifier (VC). This voting classifier takes as an input the results of the other models and creates a majority vote using the average of the predicted probabilities from the other models (Learn, 2019a). Often these can improve on the results of weak estimations from other classifiers.

3.6.2 One-Class Models

The models for this portion of the experiment were all SVMs with different kernels. Kernels are different ways of calculating dot products from two vectors and mapping them higher dimensional feature space. Effectively turning the 2-dimensional dataset

into 3-dimensions or more. This allows datasets that are not normally linearly separable to be divided by a plane and thus classified (Team, 2018; Saptashwa, 2018). A visual example can be seen in figure 3.12. In the input space the circles cannot be linearly separated but using the kernel to increase the dimension space separates the circles and allows them to be divided by a linear plane.

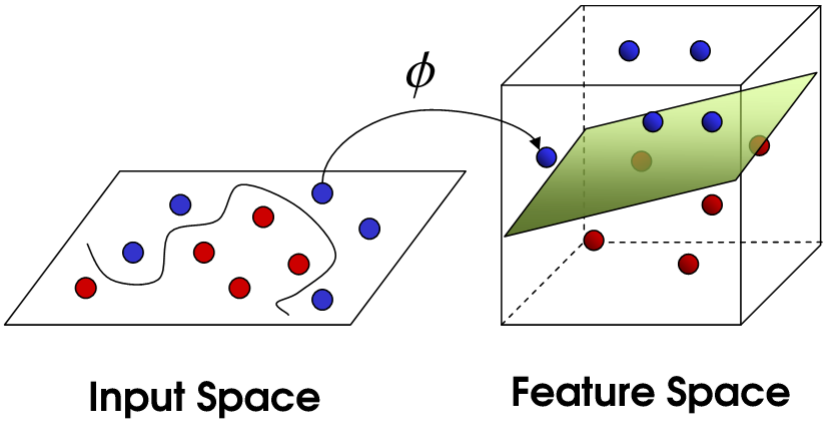

Fig. 3.12. Example of how a SVM Kernel Works (Wilimitis, 2018)

For this experiment, the chosen SVM kernels were:

- Linear
- Radial Basis Function (RBF)
- Polynomial
- Sigmoid

The goal of selecting multiple kernels was to see which, if any, could correctly classify the sessions. As with the two-class method, each phone was tested using the 70/30, 80/20, and 30/70 training splits. The next section discusses the hypotheses this research proposed.

3.7 Hypotheses

The purpose of this research is to test two hypotheses.

The first hypothesis is:

> The behavior of the owner of a phone creates unique patterns in the block space of a phone that will allow it to be uniquely identified and classified from other phones of the same make and model.

The second hypothesis is:

> The behavior of a masquerader (human or software) will be different than the owner and will leave patterns in the block structure of a smartphone that can be used to determine when a phone is not being used by the owner.

3.8 Units and Sampling

This section discusses the units being tested and the sample chosen. It also covers the variables being tested and evaluated as well as what will be considered a successful test.

3.8.1 Sample

The dataset used for this research came from the Purdue Experiment. It used a convenience sample of 34 students from Purdue University, mostly within the College of Technology, were provided Samsung Galaxy S3s for the duration of the Purdue Experiment. The subjects in this experiment were unaware of the true nature of the experiment being conducted in order to convince them to use their phones in a normal manner. If the subjects were aware of the experiment, they may have changed how they interacted with the phone and thus alter the behavior being recorded. After

the Purdue Experiment was ran, the subjects were informed of the true nature of the experiment and given a post consent form. Any subjects who wished, could withdraw from the experiment and have their data removed.

Three of the subjects did request to be removed. Which brought the total subjects down to 31. From the 110 days the experiment ran, 872 images were collected across all the phones. This made roughly one image per phone every four days. The collections were done wirelessly while the subject was on Purdue's campus. Subjects were encouraged to be on campus everyday but weekends and holidays to facilitate the successful collection runs. Granularity was not perfect however, as some phones had more sessions than others and sessions were not always collected with the same amount of time between them.

There were also five phones whose data was of undetermined origin. A few of them were used by members of the research team during the experiment to provide control and test services while the experiment was being conducted. Others were replacement phones sent out to the subjects when their phones were broken. The data for these phones were also stored in the Purdue Experiment and due to the anonymized nature of the research, the data from these phones cannot be extracted from the dataset. That being said, the behavior of the users will still have patterns based on their normal behavior and the data still was applicable to this study. One phone, identified as phone 2, had to be dropped from this research because it had only two recorded sessions. This creates a total sample size of 35 subjects.

It is important to note that the sample size of 35 subjects is significantly larger than many MDS studies with several having less than 10 subjects (Garg et al., 2006; Ahmed & Traore, 2007; Bhukya et al., 2007; Camina et al., 2011). The College of Technology at Purdue University had 3,640 students in 2013 when the Purdue Experiment was conducted (Purdue, 2016). Assuming a normal distribution, that means the study had a 95% confidence interval with a 17.8% margin of error with regards to the students in the College of Technology, now known as the Polytechnic Institute of Technology (Systems, 2012). Expanding this to all 38,788 students at

Purdue University in 2013 and the confidence interval and margin of error stay the same. This means that the results of this study can be generalized not only to students in the College of Technology, but to students at Purdue University as well.

The majority of subjects for this study were educated consumers in the age range of 18-30. Specifically, they were mostly from the College of Technology, so they were likely to have a certain mindset and comfort level when it comes to using and adapting to new technology such as smartphones. This might have caused different behavior than that of the average person in the U.S. While a larger sample would reduce the margin of error when it comes to generalizability, it is cost prohibitive to provide enough phones and service to create a larger sample. The primitive dataset being used in this study did not require the users to behave in a certain manner or use designated programs and functions which helped overcome the narrowness of the subject pool when it comes to generalizability. For a pilot study, the results are statistically significant and reliable enough to generalize to a much larger population.

3.8.2 Variables

There were two variables being examined in this study. The number of blocks that change between sessions and where those blocks are located. Each user session represents a different day in which the phone was scanned and imaged. The number of blocks that changed between sessions are representative of how much activity occurred on the phone. Where each block changed is what creates the patterns that the user's behavior is leaving behind on the phone.

3.9 Originality

The research in this paper is original for several reasons. Although it takes advantage of the PMF framework and dataset the research itself was not based on any of the results from the Purdue Experiment. In fact, the proposed study explored

a completely new space for masquerade detection instead of making refinements to methods already developed.

For example, this research went to a lower level than any of the previous methods used in masquerade detection. By examining only which blocks changed between sessions it is possible to build a user profile without needing to know any sensitive user information. This will not only lower the time needed to alert security in case of a violation, it will also maintain user privacy by keeping data encrypted until it is needed for further investigation. Analysts do not have to be exposed to user data unless a session is identified as not belonging to the user. Once a session has been flagged as aberrant, further response can be conducted based on the organization desire's, including a full forensic investigation.

Using a more primitive dataset also makes this MDS more universal than prior methods. Specific actions do not need to be taken by the user for this method to work. No matter how the user interacts with their smartphone, changes will occur in the block storage. Even malware piggybacking on a legitimate user's session will cause changes to occur and can be detected. This defeats the need to know in advance what changes are important and which features need monitoring as this research operates at the lowest common denominator.

The proposed experiment also has a novel method of generating attack sessions. Where most MDS experiments use data sessions from other users to simulate a masquerader, this experiment used actual user sessions with attacks injected into them. The reverse tethering feature allowed forensic images to be restored to a phone. This recreated the exact state the phone was in when the collection run was made. Allowing attack sessions to be generated as part of the user generated sessions instead of having to create poor simulations by switching sessions from other users. It also means multiple attacks can be conducted at the same point in time to study how they propagate on the phone. By always being able to reverse the phone back to a prior state the same session data can be used to study new attacks as they are discovered. Methods that require user sessions to be switched or faked to generate a bad session

cannot adapt to new attacks after the original user data has been collected. The framework proposed here does.

3.10 Summary

This chapter provided the framework and methodology to be used in the research study. The next chapter will provide the results of the experiment after it has been conducted.

4. ANALYSIS AND RESULTS OF THE STUDY

This section covers the analysis and results of the study. The machine-learning algorithms were trained against each phone and the model tested to see how effective they were at classifying the user sessions from their phones. The phones were numerically labeled for identification and then selected for the study using a random number generator. Phones 4, 15, 16, 27, and 32 were selected. Each phone had its sessions numbered from 1 to x with x being the last session. Randomized sessions were then chosen to be the subject of attacks by using a random number generator. The data from the experiments were then submitted to the appropriate models for each phone to find how accurately they could classify sessions conducted by a masquerader.

A session for each phone is defined as one collection run from the Purdue Experiment dataset. There was an average of 30 sessions per phone with a median of 23. The data partitions for the Samsung Galaxy S3s used in this research are 12,531 MB. Each block in PMF was set is 1 MB. This would create a feature set far too large for the study to effectively use. In order to reduce the number of features being examined from 12,531 to more reasonable number the blocks were grouped into 126 sections. Each section is 100 blocks long, except for the last which was 31 blocks in size. The number of chunks that changed in each section were added together to create a number between 0 and 99. These were then used as the feature set for each session.

When generating the two-class models, sessions for the phone being examined were considered class 1, all other sessions, class 0. From there the sessions were split into training and testing sessions. The splits were done at the standard 70/30 and 80/20 to see how the number of training samples impacted the accuracy of the models in predicting the testing samples. To further test the capabilities of the models, another set was created using a 30/70 split. Once the data was split, it was scaled using

sklearn's StandardScaler function. StandardScaler is used to center the distribution of numbers around zero (Learn, 2019c). This is used to prevent an individual feature from dominating results because it is too high or low compared to the others. Principle Component Analysis (PCA) was used to further reduce the number of features being examined. The number of components were set so that 95% of the variance could be explained (Brems, 2017). The training split data was then ready to be used to train the models. The following sections will cover the results from the machine learning models for the two-class and then the one-class algorithms.

4.1 Two-Class Results

The first hypothesis of this research is that the user of a phone will behave in a way that influences how and where the phone writes its data creating unique patterns specific to that phone. To test this hypothesis, two-class machine-learning algorithms were used to compare all the sessions for one phone from the Purdue Experiment to all the sessions for five other phones from the experiment. Pairings were done using a random number generator to select the five additional phones. If the first hypothesis is correct, then the models will be able to successfully categorize the phones as belonging to the correct class. The second hypothesis is that a masquerader will behave differently than the user and the patterns left behind will be significantly different enough that the models will be able to identify these new sessions as not belonging with other session from the phone. The following subsections discuss how the models were created and the results from running the experiments.

4.1.1 Two-Class Model Analysis

A model was generated and tested for each phone with a training/testing data split at 80/20, 70/30, and 30/70. As with other changed in parameters for the models, this was one to see if there was a significant difference in the accuracy of predictions. A receiver operating characteristic (ROC) curve was generated for each phone to show

where the maximum number of true positives occurs with minimal amounts of false positives (Narkhede, 2018).

A false positive in this case would be the model identifying a session that does not belong to the phone as belonging to it. A perfect mean ROC area under the curve (AUC) would be a 1.0. Several of the models were able to achieve this on a few phones but none consistently. Most of the time the models were able to generate AUC scores between .85 and .95 which means the number of true positives is very high compared to false positives based on only changes in the block structure. Figures 4.1 through 4.3 below are examples of the ROC curves from phone 3 from all three testing sets (See Appendix C for all of the ROC curves from the experiment). The straight dashed line in the middle of the diagrams is where a reference to the linear relationship between true and false positives, giving a baseline for how random classification would appear. The area below that line would suggest that the model is doing worse than randomly guessing and above the line signifies accurate classifications by the models.

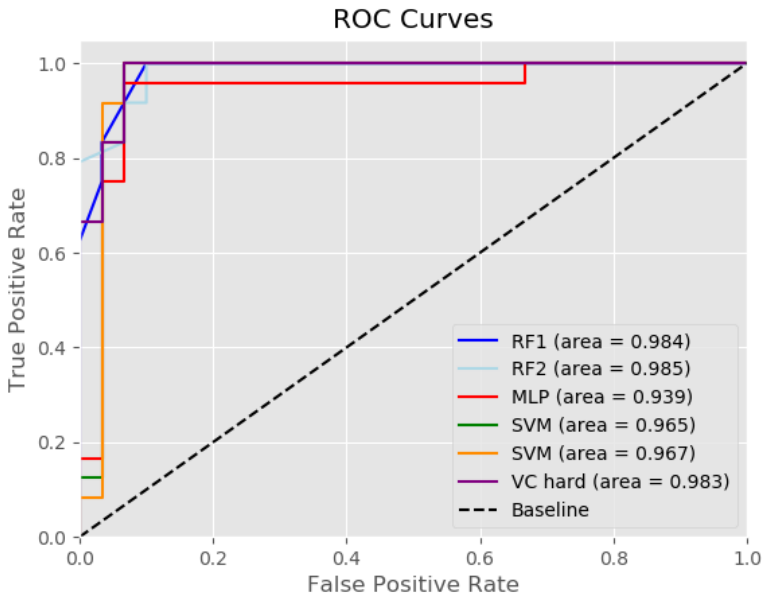

Fig. 4.1. Phone 3: 80/20 Training/Test split

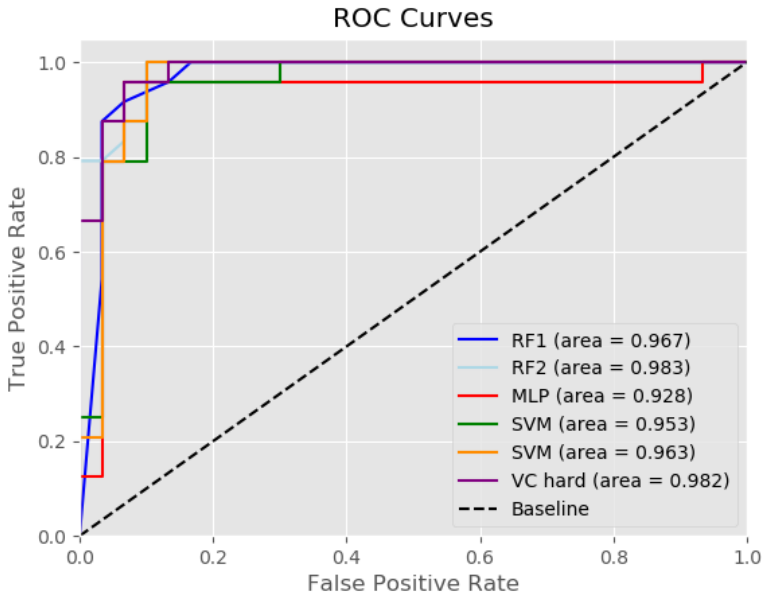

Fig. 4.2. Phone 3: 70/20 Training/Test split

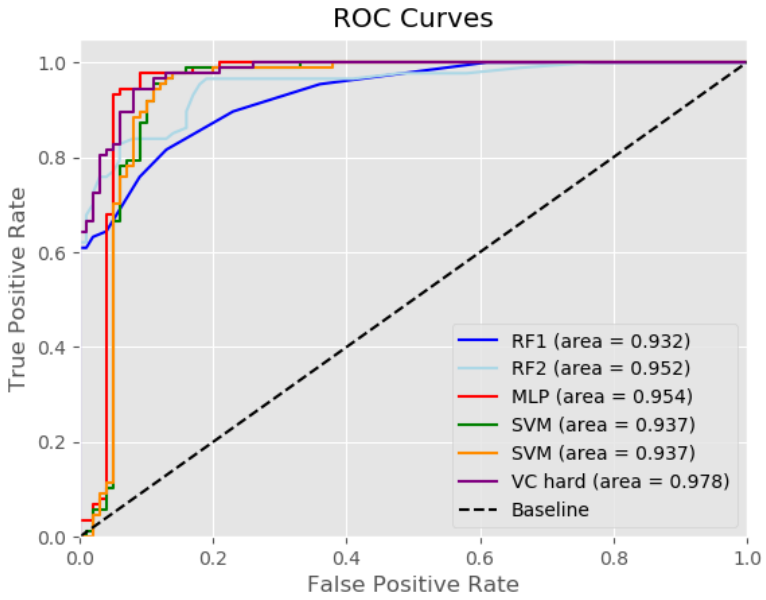

Fig. 4.3. Phone 3: 30/70 Training/Test split

As can be seen in all three example the models were capable of accurately classifying the phones with minimal false positives. Not all the phones were as easily classified as the others. While most were in low 90% range some, such as phone 12 (.826) scored significantly lower. Complete charts for all models and phones' accuracy can be found in Appendix A. The sessions from phone 12 were the most misclassified of the phones and the following ROC curves (Figures 4.4 - 4.6) were generated by the models from it. Despite being the models performing the worst on this phone, they were still able to accurately determine the correct classifications. There is more differentiation in the ROC curves from each model for phone 12 which helps show that each of the models were indeed calculating different scores and working properly.

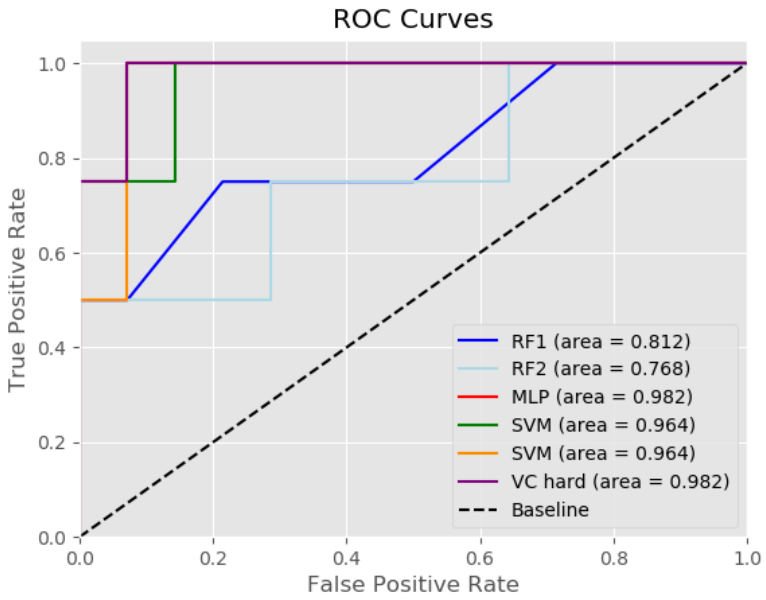

Fig. 4.4. Phone 12: 80/20 Training/Test split

A few phones, such as phone 21, also had graphs where one of the SVM2 seems to underperform, as can be seen in Figure 4.7. It also has a similar underperformace in Figure 4.9, the 70/30 test split graph. In both the 80/20 and 70/30 testing splits, several of the models appear to significantly underperform. The models were RF1 and SVM2 with 47% and 5% reported AUC. Phone 21 only had 6 sessions which

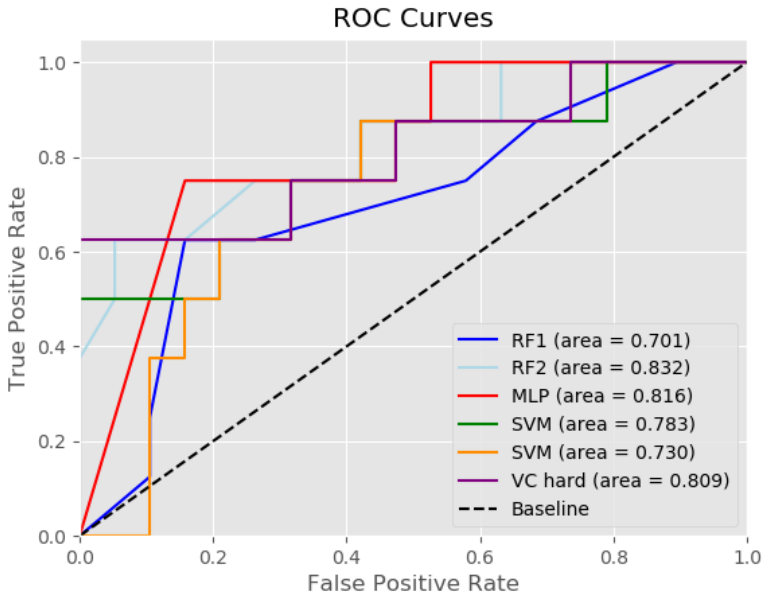

Fig. 4.5. Phone 12: 70/30 Training/Test split

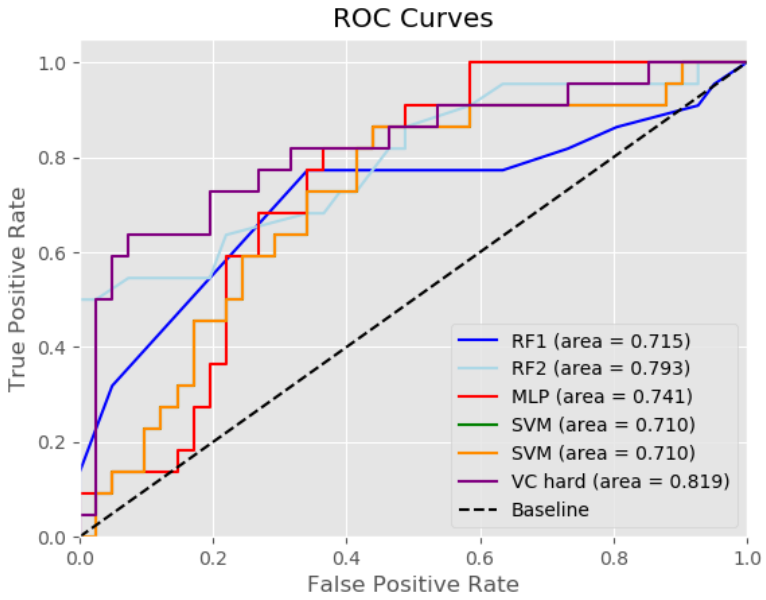

Fig. 4.6. Phone 12: 30/70 Training/Test split

would suggest the idea that a low number of sessions could be the cause of the underperforming models.

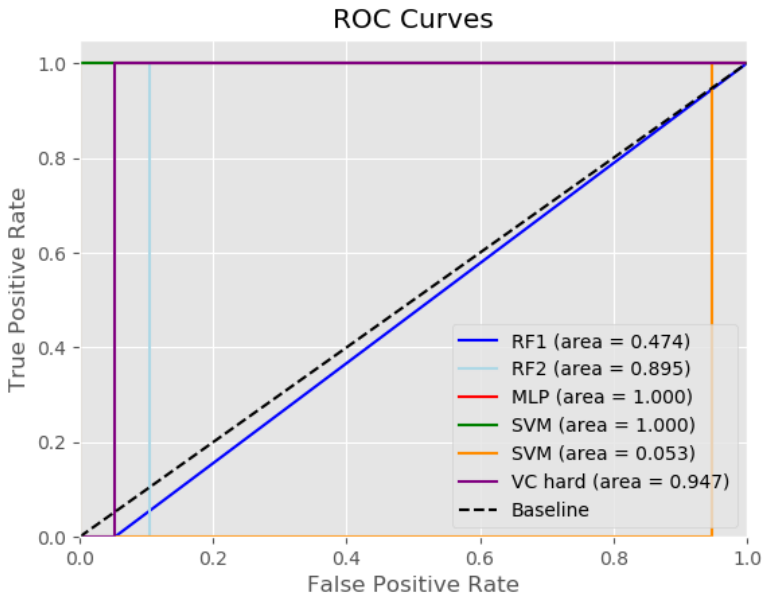

Fig. 4.7. Phone 21: 80/20 Training/Test split

Phone 24 also had a low number of sessions with only eight in total. However, it had relatively normal looking ROC graphs by comparison as seen in Figures 4.8 and 4.9 below. The low number of sessions could explain the sharp curves, but the models still performed similarly to each other and to the rest of the study. Except for SVM2 in the 70/30 test split which reports an area of only 7%. To make it even more challenging to figure out why some of the graphs are so different, the underperforming models in each phone were reported to have a high accuracy when examining the results from the models. Phone 21's RF1 and SVM2 actually had an accuracy of 95% while phone 24's were 94% and 97% accurate. This high accuracy is also supported in the graphs by the voting classifier having a high AUC of 95% (phone 21) and 97% (phone 24). If the models were truly performing as poorly as the graph indicates, the voting classifier would have had lower results. This suggests

the apparent under-performance in the graphs has more to do with the an error in the graphing of the data and not with the number of sessions.

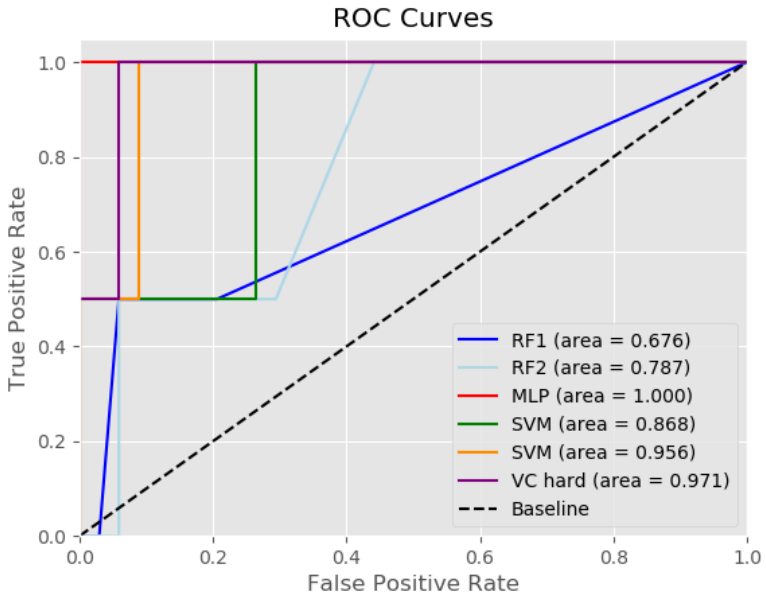

Fig. 4.8. Phone 24: 80/20 Training/Test split

Table (4.1) shows the average accuracy of predictions for each model across all the phones. Appendix A contains all the tables for each phone. These scores are useful when discussing generalities, but unlike the ROC curves, they do not show the tradeoffs between false positive and negative scores. The 80/20 training split was slightly more accurate than the 70/30 split in almost all instances, but not with much statistical significance. This was expected as there is more data being trained on and gives the models a better chance to make accurate predictions on the test data. This shows that the null for hypothesis one can be rejected. The way a phone stores its data appears to be very unique and every model on average was able to correctly classify phones sessions more than 90% of the time. The most accurate model overall based on this dataset was the Voting Classifier with 92.3% average accuracy in all three tests. This was expected as voting classifiers often outperform the classifiers from which they are comprised. The RF classifier did indeed perform

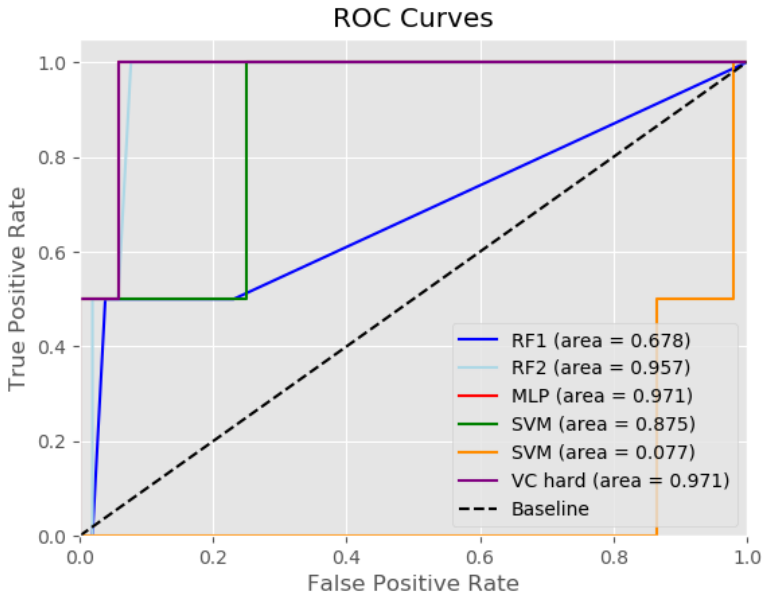

Fig. 4.9. Phone 21: 70/30 Training/Test split

better with more trees being involved which is also expected behavior. Surprisingly, though the CV SVMs performed worse than the regular SVMs and the lower C score SVM (SVM1) was more accurate than the higher one.

In the original Purdue Experiment the models were only 84% accurate at best compared to an average of 92% from this experiment. This suggests that changes in the block space are more unique to the users and a better feature than the actual data itself. This is supported by the data collected in these trial when the models are using legitimate sessions from one phone compared to other phones. The next phase of the study was designed to test how well the models could detect masquerade sessions where the experimenter made changes to the phone that would not follow the same patterns as the original user.

The same model generated by the first phase was used to examine the data from the attack sessions. Almost all the models performed worse. The average went from 92% in phase one to 37% in phase two. This provides both positive and negative feedback to the experiment. The positive feedback indicates that the attack sessions

Table 4.1.
Overall Accuracy of Classifier Models

Split	SVM1 C=10	SVM1 CV	SVM2 C=20	SVM2 CV	RF1 n=10	RF2 n=50	MLP	VC
80/20	.929	.911	.922	.906	.909	.908	.931	.937
70/30	.921	.911	.904	.906	.917	.922	.923	.923
30/70	.885	.878	.886	.890	.869	.881	.921	.905

still look more like the original phone sessions than like a different phone. The negative feedback indicates the attack sessions would get past most of the models. Most is key because the cross-validated SVMs were both able to detect the attack sessions accurately. The following tables show the results of the predictions for each model by phone.

The CV SVMs were able to on average detect the attack sessions 88% and 91% respectively. This would indicate that hypothesis two is also correct and that the behavior of a masquerader is different enough from the behavior of the owner and can be detected by some of the machine learning models. Not only can the attack sessions be found, this method of finding them appears to be as good as or better than the traditional IDS that was created in the Purdue Experiment. While successful, this method requires multiple phones to be used. The one-class tests in the next section cluster an individual's phones sessions without the need for other phones to be part of the examination.

4.2 One-Class Results

The one-class experiments were also used to test the two hypotheses, just in a different manner. In these one-class tests, all the sessions came from the phone being tested. The algorithm then clusters the sessions with ones that look the most similar being closest together. Sessions within the cluster are considered authentic user session while outliers are considered suspect and possibly belonging to a masquerader. If the first hypothesis is true, then the models should cluster the actual user sessions together with as few false positives as possible. If the second hypothesis is true, the attack sessions should be classified as outliers with as few false negatives as possible. The following subsections discuss the algorithms used and the results of the experiments.

Table 4.2.
Attack Session Results 80/20 Test

Phone	SVM1 C=10	SVM1 CV	SVM2 C=20	SVM2 CV	RF1 n=10	RF2 n=50	MLP	VC
4	0	.970	0	.977	0	.600	0	0
15	.400	.798	0	.847	.400	.400	.200	.200
16	0	.875	0	.929	.200	.200	0	0
27	0	.860	0	.851	.400	0	0	0
32	0	.952	0	.931	.400	.200	.200	.200
Average	.080	.891	0	.907	.280	.280	.080	.080

Table 4.3.
Attack Session Results 70/30 Test

Phone	SVM1 C=10	SVM1 CV	SVM2 C=20	SVM2 CV	RF1 n=10	RF2 n=50	MLP	VC
4	0	.948	0	.955	.200	.200	0	0
15	.400	.754	0	.845	.800	.400	0	.200
16	0	.934	0	.971	.200	.200	0	0
27	0	.828	0	.874	.200	0	0	0
32	0	.937	0	.928	.400	.600	.200	.400
Average	.080	.880	0	.915	.360	.280	.040	.120

Table 4.4.
Attack Session Results 30/70 Test

Phone	SVM1 C=10	SVM1 CV	SVM2 C=20	SVM2 CV	RF1 n=10	RF2 n=50	MLP	VC
4	.200	.959	.200	.967	.200	.400	.200	.200
15	.200	.719	.400	.800	.400	.400	.400	.400
16	0	.937	0	.975	0	0	0	0
27	.200	.811	0	.758	.200	.200	.200	.200
32	0	.963	0	.960	.400	.200	0	0
Average	.120	.878	.120	.892	.240	.240	.160	.160

4.2.1 One-Class Model Analysis

The results from the one-class were more varied than the two-class models. Unlike the two-class models, most of the one-class models failed. Failure in this case means the model was less accurate than random chance would dictate, less than 50%. There were several instances where a model was 100% successful while other models completely failed for an individual phone. Out of the four models used in this phase of the experiment, only the SVM using a Sigmoid kernel was able to accurately classify the user sessions. Even then it was only 68% accurate when classifying the actual user sessions. It was also the only one that successfully classified the masquerade sessions at 84% accuracy.

Testing the first hypothesis with the one-class models was done in a similar method to the two-class tests. Each phone had all its sessions split, a model was trained for it, and then tested on the remaining sessions. Table 4.5 shows how accurately each model was on average at predicting actual user sessions. The full tables for how each model predicted sessions for the phones can be found in Appendix B.

Table 4.5.
One-Class Model Average Accuracy

Test Split	Linear	RBF	Polynomial	Sigmoid
80/20	.425	.293	.170	.658
70/30	.500	.342	.203	.673
30/70	.421	.108	.192	.698
Average	.449	.248	.188	.676

For the most part, it would appear that using only sessions from an individual phone, that the first hypothesis is not true. Three out of the four SVMs were unable to do better than chance. However, the SVM using the Sigmoid kernel shows that

it is possible to successfully classify user sessions. This would suggest that the first hypothesis is true, but more experimenting is needed before it can be confirmed.

The accuracy for all the models is much lower than the two-class tests were. The most likely reason for this is that the two-class tests had five times the number of samples to work with. Investigating the idea that sample size is the issue, Table 4.6 shows all of the phones and how many sessions they had alongside the accuracy from the Sigmoid SVM's predictions. At first glance, it may appear that the results are random, with the model doing well in one test and not in another. This gives the appearance that session size may not matter, but further examination shows it does have an impact on the results. Table 4.7 shows how using only phones with more sessions improves the average accuracy of the Sigmoid SVM. When all the phones with fewer than 10 sessions are removed from the data, the accuracy raises for most of the testing sets. When the phones with 20 or fewer sessions are eliminated, the accuracy raises higher for all three testing sets. This trend continues when only phones with 30 or more sessions are used.

Table 4.6.: Sigmoid SVM Sessions to Accuracy

Phone	Sessions	Accuracy 80/20	Accuracy 70/30	Accuracy 30/70
1	12	1	.667	.286
3	125	.840	.763	.896
4	11	1	.750	.750
5	23	1	.571	.813
6	22	.800	.714	.867
7	45	.555	.429	.709
8	42	.444	.538	.966
9	50	.600	.800	.743
10	56	.636	.647	.641
11	20	.500	.330	.643

continued on next page

Table 4.6.: *continued*

Phone	Sessions	Accuracy 80/20	Accuracy 70/30	Accuracy 30/70
12	38	.750	.750	.538
13	33	.857	.500	.870
14	48	1	.933	.879
15	30	.833	.889	.905
16	41	.500	.333	.857
17	30	.833	.556	.950
18	20	.250	.333	.462
19	22	.600	.571	.733
20	18	.750	.500	.917
21	6	0	1	0
22	20	0	.167	.786
23	23	.600	.571	.500
24	8	1	1	.400
25	25	0	.500	.750
26	29	.500	.667	.769
27	19	1	.833	.842
28	27	.833	.750	.625
29	23	.800	.857	.643
30	21	.750	.667	.769
31	42	.778	.923	.500
32	42	.875	.833	.500
33	8	.500	.500	.900
34	16	.667	.200	.500
35	9	1	1	.857
36	11	.500	1	.857

Table 4.7.
Changes in Average Sigmoid SVM Accuracy

	Accuracy 80/20	Accuracy 70/30	Accuracy 30/70
All Phones	.658	.673	.698
Sessions ≥ 10	.685	.618	.714
Sessions ≥ 20	.699	.671	.714
Sessions ≥ 30	.731	.684	.766

For one-class models, the sample size is important. The phones with more samples were generally more accurately classified by the model. With at least 30 sessions, the model was 76.7% accurate. This method could probably be even more accurate if the sessions occurred more regularly and frequently. As discussed in Limitations (section 1.6), the PMF collection runs that were used as sessions for this experiment were often 24 hours or more apart. If the time between sessions was reduced and kept at more regular intervals, it is possible the inherent randomness of human behavior and thus the noise created in the drive space for the phone would also be reduced, allowing for more accuracy in the classification process. There may also be an over-fitting issue as the most accurate model was trained on the least amount of data. More sessions at more frequent intervals would need to be conducted to test this theory.

Despite most of the one-class models being relatively inaccurate at classifying the user sessions, the more important question was how well did the models predict the attack sessions. False positives, while annoying, are not nearly as bad as false negatives, where an attack session would be classified as belonging to the phone and miss going through further security. Table 4.8 shows the average accuracy of each of the SVM models at predicting the attack sessions.

Unsurprisingly, most of the models fail to correctly classify the attack sessions. The Sigmoid SVM once again was the best at classification and outperformed itself in this phase. It was more accurate at classifying the attack sessions than the actual

Table 4.8.
Average accuracy of SVMs at classifying attack

Test Split	Linear	RBF	Polynomial	Sigmoid
80/20	.440	.280	.360	.840
70/30	.560	.160	.120	.760
30/70	.520	.040	.280	.840
Average	.501	.160	.253	.813

sessions, averaging 81%. This also indicates that hypothesis two is correct. The behavior recorded during the attacks was different enough that the Sigmoid SVM was able to detect and identify the changes in the phones block storage as not belonging to the user.

4.3 Analysis Conclusions

The purpose of these experiments was to test the two hypotheses proposed by this research. The first hypothesis is that user behavior is unique and that it will cause patterns in the block structure that allow it to be identified from other phones. Hypothesis two is that sessions created by a masquerader using the phone will be different because the masquerader behaves differently.

Both the one-class and two-class models support the first hypothesis being true. The two-class models were extremely accurate at identifying the actual sessions from the sessions belonging to any other phone. The average accuracy for the two-class models was over 90%. The Sigmoid SVM from the one-class tests was also able to successfully identify the actual user sessions. For the first hypothesis to be true, user behavior would have to create enough differences in the underlying block structure of the phone. This would allow them to be correctly classified by the machine learning models. While the current study does not provide conclusive proof, it does show that

the models appear to be able to classify sessions correctly, which is why both sets of models support hypothesis one being true.

Both the one-class and two-class class models also support the second hypothesis being true. For hypothesis two to be correct, the models would have to be able to tell the difference between the user session and the attack sessions. The process of creating the attack sessions is inherently different than the behavior of the the phone's actual user. If hypothesis two is true, these differences should create a different pattern in the block structure and be detectable by the models. Both of the CV SVMs and the Sigmoid SVM were able to accurately classify the attack sessions as not belonging to the phone. The two cross validated SVMs were both about 90% accurate at classifying the attack sessions. The Sigmoid SVM was less accurate, but was still able to correctly identify the attack sessions 80% of the time. The high accuracy in both types of models suggests that hypothesis two is also true.

Not all the models were successful though and even the best one-class model (Sigmoid SVM) was not as accurate as desired. There are many factors that could be responsible for the lower accuracy. One that was already shown to be a factor is the low number of sessions for some of the phones. In section 4.2.1 it was shown that having more sessions improved the accuracy of the models. Phones that had at least 30 sessions were generally more accurately predicted than phones with fewer sessions. Some phones with a low number of sessions were still accurately classified for example: Phone 19 only had 18 sessions, but it was still accurately classified by the two-class models and the one-class Sigmoid SVM. Phone 7 on the other hand had 45 sessions and was not accurately classified by the Sigmoid SVM. This means the number of sessions is not the only import factor.

Frequency is also an issue. PMF scanned the phones for changed sections of device storage every four hours, but the collection of the session data was only once every 24 hours at best. Often three or more days could pass before the data was collected and a new session made. This potentially left too much time between sessions and allowed for too much noise in the data. Phones such with a higher than average number of

sessions, such as phone 7, that were poorly classified might have better results if the time between collections was reduced. Increased frequency means less time for the randomness of human behavior to create noise in the storage space of the phone.

Two of the attacks involved actions that could be taken by the user during the course of normal activity. These attacks were the deletion and installation of an app. When sessions with these attacks were ran through the machine learning models, the CV SVMs were able to identify them as attacks. That is a positive result, but could suggest that any action the user takes that installs or deletes an app will be determined as aberrant behavior. It is certainly possible that if enough blocks in a session change that the models will always predict the session as an attack. It is also likely that the subjects in the Purdue Experiment did install and delete apps. These actions could be what caused some user sessions to be incorrectly predicted. It is also possible that sessions with these actions were correctly predicted. Without looking at the actual user data, it is not possible to know. Future research should include protocols for this, but it is outside of the scope of this research.

This was a pilot study that utilized machine learning. Before working with the various models and trying to utilize them with the data, it is nearly impossible to know which ones will work beforehand. It is entirely possible that alternative models and machine learning algorithms will work better than the ones tested in this experiment. With the one-class methods in particular, there are many other models and kernels that could potentially be better at classification and prediction.

While not all the models were successful and not all the variation in classification can be explained by this study, the overall objectives of this experiment were achieved. For a pilot study, this research was very successful and showed results that supported both hypothesizes. The next section of this paper discusses the conclusions and ramifications of this research and what it means to cyber forensics, security, and privacy.

5. CONCLUSIONS

The purpose of this research was to answer the question, how effective can assessing the block structure of a phone be at detecting masqueraders, malware, and exploitation. To study this, two hypothesis were tested. First, that user behavior creates unique patterns in the block structure of the phone that can be uniquely identified and classified. The second is that the behavior of a masquerader using the phone is different than the owner and will leave different patterns behind that can be The results of this study show that the underlying block structure of a phone does change in a manner unique to the phone and it's user. The patterns created in the block structure by the user interacting with their phone are able to be recognized by machine learning algorithms. The models created by the algorithms were then able to correctly classify legitimate sessions as belonging to the user. These models were then tested using attack sessions created from legitimate session data. Several of the models were able to correctly identify the attack sessions as not belonging to the legitimate user. The results of this study indicate that the hypothesis are correct and that this method of masquerade detection is effective. This has important implications for cyber forensics, security, and user privacy. Despite the success of this study, more research needs done into this method of masquerade detection to verify the results and efficacy.

The Purdue Experiment was considered a successful test using PMF to create a traditional MDS for smartphones. Using a similar two-class Random Forest model, the Purdue Experiment was able to correctly classify sessions from phones 84% of the time (Guido et al., 2016a). To intentionally allow a comparison to a known good value, two different random forest models were tested in this study, along with several other model types. The random forest models tested in this study were nearly identical to each other, with only the n variable being different. The n variable sets how many

trees the model uses, in this case it was 10 and 50 in order to test how a larger number of trees would affect the results. Generally speaking, the more trees, the more accurate the model at an increase of computational complexity. The random forest models in this study best average accuracy was .917 and .922 in the 70/30 training/test split respectively. This was a surprisingly high result as no personal user data was examined in this research. Even more surprising, the random forest models were not the most accurate models tested in this study.

The most accurate of the two-class models at was the VC in the 80/20 training split with an average accuracy of .937. This is a little misleading as the VC is a composite model based on the accuracy of all the other models combined. The next most accurate model was the MLP with an average accuracy of .931 in the 80/20 training/test split. Both of these results are almost 10% more accurate than the random forest model from the original Purdue Experiment.

The two-class tests required the comparison of sessions from one phone to sessions from other phones. To test how similar sessions from a single phone were to each other, one-class models were also tested. Most of these models were not accurate with the only one that did better than random chance being the Sigmoid SVM. The Sigmoid SVM's best accuracy was .698 in the 30/70 training/test split. This accuracy was easily increased to .766 by removing phones that had fewer than 30 sessions. This shows two items of interest. The first being that the one-class models are more sensitive to sample size than the two-class. The second is that there may be some over fitting issues as the accuracy was highest when the training data was at the lowest.

The purpose of conducting both the one and two-class tests using only valid session data was to test the first hypothesis and show that changes in the block structure can be used to create user profiles. Without being able to successfully classify user sessions there would no point in testing the second hypothesis. Testing the attack sessions changed which of the two-class models were the most effective. When testing the models using the attack sessions, only the CV SVMs were accurate at 88% and

91% respectively. While these were not the most accurate models for test the first hypothesis they were still over 90% accurate. They maintained that accuracy when identifying attacks created from legitimate user data. The Sigmoid SVM was still the only one-class model that could accurately classify the attack sessions. Here the accuracy actually rose to an average of .813. The high accuracy of these models support hypothesis 2 being true.

If the Purdue Experiment was considered successful at 84% then this experiment was successful as well. The Random Forest models for this study were even more accurate when classifying legitimate user sessions at 91.7% and 92.2% accuracy. Unfortunately, the RF models were not able to accurately classify the attack sessions at 29.3% and 26.7%. The best at classifying both legitimate sessions and the attack sessions were the CV SVMs. These SVMs were 87.8& and 89.0% accurate when predicting legitimate sessions. They were just as accurate when predicting if the attack sessions at 88.3% and 90.5%. The Sigmoid SVM was also fairly accurate at 67.6% and increased to 76.6% when the amount of sessions was accounted for. It was also able to correctly classify the attack sessions 81.3% of the time. Considering it was unknown if any of the models would be any more accurate than random chance, these results were very promising.

It would appear that the answer to the research question is that examination and analysis of the underlying block structure of a smartphone is highly effective at detecting masqueraders, malware, and exploitation. This study had several limitations that should to be addressed before it can be declared that the hypotheses are proven facts. As a pilot study there were many limitations that restrict the how definitively the results can be taken. The biggest limitation is that the dataset used is from the Purdue Experiment and was not designed with this study in mind. The sessions were not collected with the regularity and granularity that would have been ideal for this study. There were also not as many sessions as would be desired. Greater granularity, regularity, and recorded sessions would allow for better analysis of the data and could provide greater insights. The analysis in 4.2.1 has already exposed that the number

of sessions has a large impact on the accuracy of some of the models. There was also only one model of smartphone and version of the Android operating system involved in this study. It is possible that other phones and operating systems could behave differently enough that it would invalidate the findings of this research, but social cognitive theory would suggest otherwise. User behavior also changes over time and the longer the experiment the more likely these changes would be recorded. Changes in a users behavior could introduce noise into the data that would make classification of sessions difficult. The Purdue Experiment was run over a course of 109 days, which is a significantly longer time frame than most other research into masquerade detection. This provided a decent amount of time for users to alter their behavior, but cannot account for everything.

Another limitation to this research was that user data was not allowed to be decrypted and examined. This means that legitimate sessions that were misclassified cannot be further investigated. The false positives, sessions classified as being abberant when they aren't, are a waste of time for investigators. The false negatives, sessions classified as belonging to the user when they do not, are even worse as it means an attack has gone undetected. If it was known what caused the prediction, the data could be accounted for to make the models more accurate. Even without this knowledge though, the accuracy of the models was high, especially for a pilot study.

Other weaknesses come from the study itself. One of the biggest weaknesses was caused by time and resource constraints and can be easily rectified by additional testing. That shortcoming is that only two malware samples were used in the this study. Additional samples of malware can be used in future studies to examine if the findings in this research still hold true. The other big weakness is that all of the attacks had to be conducted by first deleting the security files so that the researcher could access the phone. This can have the unintended consequence of all having all of the models detect that and not the actual attack. In all 5 of the studied attacks there are examples where each model failed to predict the session where only deletion

of the security files occurred as an illegitimate session while still correctly predicting the other attacks. Additional research could be done to verify it is the attack that is being detected and not only the deletion of security files,

Despite the weaknesses in this study, the results strongly support both hypothesis. The implications this has for cyber forensics, security, user privacy, masquerade detection, and even psychological theory should not be underestimated. Both classes of models showed that the patterns created in the block structure of the phone by masquerader is different than those of the legitimate user's behavior. The differences seem to be caused by the masquerader having different behavior. The literature review for this research suggested that user behavior should influence how the block structure of the phone is utilized, but until this study was done, there was no research that directly showed this to be the case. As discussed in section 2.2, reciprocal determinism theorizes that personality and behavior impact the environment, in this case the storage space of the smartphone (Bandura, 1989). While not emphatically proving it, the results of this research support social cognitive theory and the idea of reciprocal determinism. Even though each phone in the study was the same model, with the same operating system, and identical software, the block structure of each phone significantly diverged. The primary catalyst of the change in the block structure of the phone was the user interacting with it. Those differences were accurately classified by the machine learning models, especially in the two-class tests which averages 92% accuracy in classifying user sessions.

Masquerade detection research can also benefit greatly from this research. Attack sessions for many MDS research are sessions that are swapped from other users involved in the study. This can create room for bias as normal behavior for each user can be significantly different. The MDS instead of detecting actual attacks is instead detecting differences in between legitimate users. This research used reverse imaging to restore forensic images back onto a phone, recreating the exact condition the phone was in as it was being used by the subject. Putting the phone back in this state allowed attacks to be conducted directly against the phone as if it were still in

the subjects hands. When the phone was forensically imaged again, the new image is a legitimate user session with the changes cause by the attack recorded within it. Other research can use this method to create attack session that are more accurate than those created by session swapping.

As smartphones are further integrated into people's lives, the protection of privacy is all the more important and shouldn't be sacrificed in the name of security. This method of MDS has an added benefit in that it also protects user privacy. Unlike other MDS that rely on substantial amounts of sensitive user information, the methods employed in this research were conducted at a lower layer and does not rely on knowing any specifics about the user. Not only does this protect the user's privacy, it eases the burden on enterprise security as sensitive data is not exposed by default operations or in the event of a breach and compromise.

The models can also be used to show which blocks caused a session to be classified as illegitimate. This can be used to extract and decrypt only the necessary data from the image, allowing the remaining user data to stay private. If a full forensic investigation is deemed necessary the entire image can be restored, but this method means the first look does not expose all of the potentially private and sensitive information on the phone. It also means that examiners can spend less time on initial investigations as the amount of data they must sift through is greatly reduced. Another security advantage is that this method is based on heuristics and not signatures. Any attack that alters how the phone behaves can potentially be detected. Not only could zero days be detected, but the time to detection for attacks can be reduced depending on the time the occurs between sessions. This could prevent greater security incidents from occurring.

One of the biggest security threats to an enterprise is an insider threat. While most of this research focused on attacks against the user and their phone, this is an attack where a legitimate user behaves in a manner to cause harm to the enterprise. If the insider threat in this case is uses their smartphone as part of their attack, the shift in behavior could potentially be detected by this method. Suddenly transferring

large amounts of data or connecting to systems and networks that the phone does not normally connect to could reveal an insider threat in action. This is another way to use the changes in block structure to provide security.

After an incident occurs, often a forensic investigation is required. In cyber forensics, one of the most difficult questions to answer is, who was using the device? Attribution has always been challenging as artifacts on a computer or smartphone can show what and how, but provide little evidence to say specifically who was using the device. Without some external form of monitoring such as a security camera, most of the evidence as to who is circumstantial. Even if a phone remained in the owner's possession the entire time, it does not necessarily prove they were in control of the phone. Malware of all types can run without the user ever being aware of it. This method, however, is based on user behavior. When implemented it can be used provide evidence of who or what was using the phone at the time in question. Behavior that is indicative of the user would provide direct attribution that adds weight to all the other evidence discovered on the phone.

This is incredibly important to cyber forensics as it can be used to confirm or invalidate theories about what actually happened. An example of this is the "malware did it" defense. While malware can certainly be responsible for taking nefarious actions the system used here would be able to establish if the behavior fell in line with normal user behavior or if a masquerader. This could provide clarity to issues where it argued that someone else or malware took the action and not the user. If malware was really acting at that time, the session should be seen as aberrant and classified by the models as not belonging to the user. This could prove helpful in many types cases.

The purpose of forensics is to reveal the truth about what occurred. Often this can be murky and hard to determine. An example discussed in section 1.3 is the *U.S. v Moreland* (2011) where it could not be determined who was actually using the computer when illegal pictures were downloaded. If a system similar to the one in this study had been implemented it may have been possible to see who was using

the computer based on their behavior. It becomes much harder to convince a jury that someone else/malware is responsible when the behavior is shown to be indicative of the user. At the same time can provide exculpatory evidence if the behavior is deemed aberrant by the models.

This method also allows for a unique perspective when conducting an investigation. In most investigations, the forensic image is made once and only after an incident has occurred. This method here would allow for examinations over several time frames. This could help provide a clearer picture of how and what occurred. An attack that occurs over several sessions would show the order of how that attack progressed. This could be used to create more accurate timelines and establish a better understanding of how the attack occurred. Security research would also benefit by being able to examine the tactics, techniques and procedures of the attack.

Perhaps the most important result of this research is that it shows that new and innovative methods for security and forensics can be successful. Mobile phones, while technically computers, are different than traditional desk/laptops that have dominated enterprise computing. They are far more personal and integrated into their user's life and that needs to be respected. Traditional methods, while effective, are not necessarily the most appropriate. They are often resource intensive and can decrease performance or cause other operational impacts that have a deleterious impact on user experience. The methodology in this research is fast and designed to take advantage of the unique properties of a mobile phone, without placing a burden on the phone's resources or creating a negative user experience.

This was a pilot study and more research will need to be conducted before it can be claimed to better than current methods, but it shows that there is room for improvement and innovation. It resulted in strong support of both hypothesis with the best model being able to identify between user sessions and attacks 91% of the time. Suggesting that changes in the underlying block structure of a smartphone are an effective means of identifying between a legitimate user and a masquerader. While there are limitations and weaknesses to this study, the results justify continuing

research into this method of masquerader detection. Once this method has been validated and had time to mature, it can be used to improve privacy, security, and forensics.

6. FUTURE WORK

Being a pilot study means that more research needs to be conducted before declaring this method viable for deployment in an enterprise environment. There would need to be longer studies with more frequent and regular user sessions created. Different models of machine learning would need to be tested as well as alternate methods such as including a moving window to adjust for changes in user behavior over time. As this method of masquerade detection holds promise, it is possible to develop other novel methods of detection using a similar method as well.

One of the limitations of this study is that all the phones involved in it were Samsung Galaxy S3s. Future research should test multiple models of phones and versions of Android to confirm that the results of this study are generalizable to all Android smart phones. There was also a limitation in the granularity of the sessions for this experiment. The Purdue Experiment often had several days go between collection runs, creating sessions that could introduce more noise and prevent proper classification. It would be beneficial to have a dataset with more frequent sessions to study the impact of shorter times between the collection runs. This could result in more accurate models that are able to detect malicious activity faster.

It would also be beneficial to test more attacks and malware to confirm that different activities can be caught by this method of detection. This research only examined five different attack vectors in order to test the feasibility of this style of masquerade detection. As there are a wide variety of attacks that can target a smart phone, many more attacks will need to be tested before this method can be considered a general detector and used in the wild. The high levels of accuracy in the two-class and Sigmoid SVM suggest that this method is worth pursuing and could be useful in enterprise environments.

There is also interest in seeing if instead of user behavior, application behavior can be profiled across multiple phones. This would involve installing an application on multiple phones and using PMF to create sessions. The goal would then be to see if the application changes the same number of blocks at the same time and frequency. If so, it may be possible to profile behaviors as belonging to an application or class of applications. This could also be applied to malware research and provide another behavioral heuristic for malware detection.

BIBLIOGRAPHY

AFP (2013). How smartphones are on the verge of taking over the world. Technical report, AFP Relaxnews.

Ahmed, A. and Traore, I. (2007). A new bio-metric technology based on mouse dynamics.

Anderson, M. (2017). Many smartphone owners don't take steps to secure their devices.

ARGUS (2019). Argus cyber security lab.

Bach, O. (2015). Mobile malware threats in 2015: Fraudsters are still two steps ahead.

Bandura, A. (1986). Social foundations of thought and action: A social cognitive theory.

Bandura, A. (1988). Organisational applications of social cognitive theory. *Australian journal of management*, 13(2):265–302.

Bandura, A. (1989). Social cognitive theory. *Annals of child development*, 6:1–60.

Benioff, M. and Lazowska, E. (2005). Cyber security: A crisis of prioritization. dtic document.

Bhukya, W., Kommuru, S., and Negi, A. (2007). Masquerade detection based upon gui user profiling in linux systems. *Annual Asian Computing Science Conference*.

Boshell, B. (2017). Average app file size: Data for android and ios mobile apps.

Brems, M. (2017). A one-stop shop for principal component analysis.

Camina, B., Monroy, R., Trejo, L., and Sanchez, E. (2011). Towards building a masquerade detection method based on user file system navigation. mexican international conference on artificial intelligence (pp. 174-186).

Chebyshev, V. (2019). Mobile malware evolution 2018.

Chebyshev, V., Sinitsyn, F., Parinov, D., Larin, B., Kupreev, O., and Lopatin, E. (2019). IT threat evolution q2 2019. statistics.

Clarke, N. and Furnell, S. (2007). Advanced user authentication for mobile devices. *Computers & Security*, 26(2):109–119.

ComScore (2013). Comscore reports august 2013 u. s. smartphone subscriber market share.

Coull, S. and Szymanski, B. (2008). Sequence alignment for masquerade detection. *Computational Statistics & Data Analysis*, 52(8):4116–4131.

CTIA (2016). Wireless quick facts.

CTIA (2019a). 2019 annual survey highlights.

CTIA (2019b). The state of wireless-2018.

CyberEdge (2015). 2015 cyberthreat defense report. cyberedge group. Technical report, CyberEdge Group.

Dash, S., Reddy, K., and Pujari, A. (2005). Episode based masquerade detection. *Information Systems and Security*, pages 251–262.

Deitrick, C. (2015). Smartphone thefts drop as kill switch usage grows but android users are still waiting for the technology.

Garg, A., Rahalkar, R., Upadhyaya, S., and Kwiat, K. (2006). Profiling users in gui based systems for masquerade detection.

Garnaeva, M., Chebysgev, V., Makrushin, D., Unuchek, R., and Ivanoc, A. (2014). Kaspersky security bulletin 2014: Overall statistics for 2014.

Gibbs, N. (2012). Your life is full mobile. we walk, talk, and sleep with our phones. but are we more or less connected.

Goldstein, J. (2019). What are insider threats and how can you mitigate them?

Grance, T., Hash, J., Peck, S., Smith, J., and Korow-Diks, K. (2002). Security guide for interconnecting information technology systems, special publication 800-47.

Grover, J. (2013). Apple trademarks "there's an app for that".

Grover, J. (2015). Mobile forensics: Using traditional techniques to solve current problems.

Guido, M., Brooks, M., Grover, J., Katz, E., Ondercek, J., Rogers, M., and Sharpe, L. (2016a). Generating a corpus of mobile forensic images for masquerading user experimentation.

Guido, M., Buttner, J., and Grover, J. (2016b). Rapid differential forensic imaging of mobile devices. *Digital Investigations*, 18:46–54.

Guido, M., Ondricek, J., Grover, J., Wilburn, D., Nguyen, T., and Hunt, A. (2013). Automated identification of installed malicious android applications. *Digital Investigations*, 10:96–104.

Gupta, S. (2012). Using computer interaction behavior to differentiate between users to assist with digital investigations.

Handbook, F. (2012). Locard's exchange principle.

Hecht-Nielsen, R. (1989). Theory of the back-propagation neural network.

Hoyle, C., Speechley, N., and Burnett, R. (2016). The impact of being wrongly accused of abuse in occupations of trust: Victims' voices.

Huang, L. and Stamp, M. (2011). Masquerade detection using profile hidden markov models. *Computers & Security*, 30(8):732–747.

HubPages (2016). Most common attacks on mobile phones.

Hungenberg, T. and Eckert, M. (2019). Inetsim: Internet services simulation suite.

IngramMicroAdvisor (2016). 23 byod statistics you should be familiar with.

I.T.U. (2013). Ict facts and figures.

Kalis, B. (2014). Research: How many apps are in each app store?

Kessem, L. (2016). Mobile malware competition rises in underground markets.

Lane, T. (2000). Machine learning techniques for the computer security domain of anomaly detection.

Lane, T. and Brodley, C. (2003). An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, 51(1):73–107.

Learn, S. (2019a). Ensemble methods.

Learn, S. (2019b). Neural networl mlp classifier.

Learn, S. (2019c). Preprocessing standard scalar.

Learn, S. (2019d). Rbf svm parameters.

Learn, S. (2020a). One-class svm with non-linear kernel (rbf).

Learn, S. (2020b). Support vector machines.

Locard, E. (1934). La police et les méthodes scientifiques.

Lohrum, M. (2014). Imaging an android device.

Lookout (2014). 2-14 mobile threat report. lookout. Technical report, Lookout.

Lookout (2016). A mobile data breach could cost an enterprise $26.4 million.

Lookout (2017). Straight to costs: What a mobile data breach will do to your bottom line.

Lunt, T. (1990). Ides: An intelligent system for detecting intruders.

Mantyjarvi, J., Lindholm, M., Vildjiounaite, E., Makela, S., and Ailsto, H. (2005). Identifying users of portable devices from gait pattern with accelerometers.

Martin, L. (2016). The cyber kill chain.

Maxion, R. and Townsend, T. (2002). Masquerade detection using truncated command lines. *Dependable Systems and Networks*, (pp.:219–228.

Mazhelis, O. and Puuronen, S. (2005). Characteristics and measures for mobile-masquerader detection.

Mazhelis, O. and Puuronen, S. (2007a). Comparing classifier combining techniques for mobile-masquerader detection.

Mazhelis, O. and Puuronen, S. (2007b). A framework for behavior-based detection of user substitution in a mobile context.

Michael, M. (2016). These were the top 10 android threats in 2015 - plus, what to expect in 2106.

Miettinen, M. and Halonen, P. (2006). Host-based intrusion detection for advanced mobile devices. *20th International Conference on Advanced Information Networking and Applications*, 1:72–76.

MITRE (2015). Adversarial tactics, techniques, and common knowledge.

MITRE (2018). Att&ck 101.

Morgan, S. (2016). Cyber crime costs projected to reach $2 trillion y 2019.

Narkhede, S. (2018). Understanding auc - roc curve.

Nielsen (2014). Smartphones: So many apps, so much time.

NIST (2016). Mobile threat catalogue.

Osborne, C. (2015). Most comapnies take over si months to detect data breaches.

Phillips, T. (2017). Incredible 183 phoens stolen every day - how to keep yours safe and the ones thieves target the most.

Posadas, M., Mex-Perera, C., Monroy, R., and Nolazco-Flores, J. (2006). Hybrid method for detecting masqueraders using session folding and hidden markov models.

Purdue (2016). Enrollment by college, school and by student level. Technical report, Purdue University.

Pusara, M. and Brodley, C. (2004). User re-authentication via mouse movements.

Rogers, M., Scarborough, K., Frakes, K., and San Martin, C. (2007). Survey of law enforcement perception regarding digital evidence.

Ruggiero, P. and Foote, J. (2011). Cyber threats to mobile phones.

Ryan, J., Lin, M.-J., and Miikkulainen, R. (1998). Intrusion detection with neural networks.

Sager, T. (2014). Killing advanced threates in their tracks: An intelligent approach to attack prevention.

Salem, M. (2012). Towards effective masquerade attack detection.

Salem, M. and Stolfo, S. (2009). Masquerade attack detection using a search-behavior modeling approach.

Salem, M. and Stolfo, S. (2011). Modeling user search behavior for masquerade detection.

Samani, R., Davis, G., and McAfee (2019). Mcafee mobile threat report q1, 2019.

Samfat, D. and Molva, R. (1997). Idamn: An intrusion detection architecture for mobile networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1373–1380.

Saptashwa (2018). Support vector machine: Kernel trick; mercer's theorem.

Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., and Vardi, Y. (2001). Computer intrusion: Detecting masquerades. *Statistical Science*, 58(74).

Shabtai, A., Kanonov, U., and Elovici, Y. (2010). Intrusion detection for movile devices using the knowledge-based, temporal abstraction method. *Journal of Systems and Software*, 83(8):1524–1537.

Shi, E., Niu, Y., Jakobsson, M., and Chow, R. (2010). Implicit authentication through learning user behavior.

Skymind (2019). Beginner's guide to multilayer perceptrons.

Snell, B. (2016). Mobile threat report: Whats on the horizon for 2016.

Solnik, M. and Blanchard, M. (2014). Cellular exploitation on a global scale: The rise and fall of the control protocol.

Spreitzenbarth (2016). Current android malware.

Sun, B., Yu, F., Wu, k., and Leung, V. (2004). Mobility-based anomaly detection in cellular mobile networks.

Svajcer, V. (2014). Sophos mobile security threat report. sophos.

Symantec (2016). 2016 internet security threat report.

Symantec (2019). 2019 internet security threat report.

Systems, C. R. (2012). Sample size calculator.

Tapellini, D. (2014). Smartphone thefts rose to 3.1 million in 2013.

Team, D. (2018). Kernel functions - introduction to svm kernel & examples.

Team, F.-L. T. R. (2016). Ascending the ranks: The brazilian cybercriminal underground in 2015.

Tech-Faq (2015). Known ciphertext attack.

Techopedia (2020). Mobile phone.

U.S. (2011). Moreland, 09-60566 (fifth circuit court of appeals dec 14.

Vanja, S. (2014). Sophos mobile security threat report.

Vijayan, J. (2016). Android security round-up: Mobile malware's surprisingly high cost, data-leaking app risks and more.

Wainwright, A. (2016). 10 stats that show it's time to prepare for byod.

Wilimitis, D. (2018). The kernel trick in support vector classification.

Yazji, S., Scheuermann, P., Dick, R., Trajcevski, G., and Jin, R. (2014). Efficient location aware intrusion detection to protect mobile devices. *Personal and Ubiquitous Computing*, 18(1):143–162.

Yui, T. (2019). Understanding random forest.

Yung, K. (2004). Using self-consistent niave-bayes to detect masquerades.

Zachary, B. (2014). There are officially more mobile devices than people in the world.

Zorz, Z. (2016). Android spyware targets business executives.

APPENDICES

A. TWO-CLASS MODELS - AVERAGE RESULTS

Table A.1.: Two-Class 80/20 Test Results

Phone	SVM1	SVM1 CV	SVM2	SVM2 CV	RF1	RF2	ML	VC
1	0.961	0.951	0.941	0.907	0.922	0.922	0.961	0.961
3	0.963	0.953	0.963	0.93	0.926	0.926	0.944	0.963
4	1	0.985	1	0.977	1	1	0.970	1
5	0.971	0.914	0.914	0.914	0.886	0.886	0.886	0.886
6	0.939	0.892	0.879	0.867	0.909	0.939	0.879	0.939
7	0.866	0.920	0.857	0.928	0.886	0.914	0.943	0.914
8	0.812	0.881	0.844	0.901	0.844	0.812	0.906	0.844
9	0.963	0.917	0.889	0.877	0.741	0.778	0.926	0.926
10	0.907	0.959	0.907	0.93	0.907	0.907	0.907	0.930
11	0.977	0.924	0.930	0.921	0.930	0.930	0.953	0.930
12	0.889	0.721	0.944	0.720	0.778	0.778	0.889	0.889
13	0.862	0.931	0.897	0.906	0.897	0.862	0.862	0.897
14	0.879	0.916	0.970	0.879	0.818	0.879	1	0.970
15	0.952	0.798	0.952	0.847	0.952	0.952	0.952	0.952
16	0.818	0.908	0.818	0.929	0.909	0.818	0.773	0.864
17	0.973	0.905	1	0.911	0.892	0.892	0.973	0.946
18	1	0.957	1	0.979	1	1	1	1
19	0.966	0.877	0.931	0.928	0.966	0.931	0.897	0.931
20	1	0.973	1	0.951	0.957	0.957	1	0.957
21	0.950	0.934	0.950	0.926	0.950	0.950	0.950	0.950
22	0.872	0.923	0.872	0.967	0.897	0.923	0.923	0.949
23	0.938	0.872	0.875	0.914	0.844	0.844	0.969	0.906
24	0.944	0.937	0.972	0.924	0.944	0.944	0.972	0.944
25	0.958	0.935	0.917	0.871	0.958	0.958	0.917	0.958
26	0.933	0.948	0.867	0.920	0.900	0.933	0.867	0.900

continued on next page

Table A.1.: Two-Class 80/20 Test Results

Phone	**SVM1**	**SVM1 CV**	**SVM2**	**SVM2 CV**	**RF1**	**RF2**	**ML**	**VC**
27	0.880	0.910	0.920	0.851	0.920	0.880	0.960	0.960
28	0.881	0.898	0.905	0.912	0.929	0.905	0.905	0.952
29	0.906	0.929	0.906	0.922	0.812	0.844	0.875	0.875
30	0.880	0.938	0.880	0.814	0.960	0.960	0.960	0.960
31	0.800	0.874	0.800	0.839	0.867	0.767	0.867	0.900
32	0.944	0.917	0.944	0.931	0.833	0.889	0.944	0.972
33	1	0.936	0.964	0.945	0.964	0.964	0.893	0.929
34	0.969	0.896	0.906	0.860	1	0.969	1	0.969
35	0.981	0.944	0.963	0.940	0.963	0.981	0.963	0.981
36	0.967	0.814	1	0.958	0.967	1	1	1
Average	0.928	0.911	0.922	0.905	0.909	0.908	0.931	0.937
Attack	**Sessions**							
4	0	0.970	0	0.977	0	0.600	0	0
15	0.400	0.798	0	0.847	0.400	0.400	0.200	0.200
16	0	0.875	0	0.929	0.200	0.200	0	0
27	0	0.860	0	0.851	0.400	0	0	0
32	0	0.952	0	0.931	0.400	0.200	0.200	0.200
Average	0.0800	0.891	0	0.907	0.280	0.280	0.080	0.080

Table A.2.: Two-Class 70/30 Test Results

Phone	**SVM1**	**SVM1 CV**	**SVM2**	**SVM2 CV**	**RF1**	**RF2**	**ML**	**VC**
1	0.935	0.938	0.935	0.956	0.948	0.948	0.961	0.974
3	0.938	0.898	0.938	0.926	0.901	0.938	0.938	0.938
4	1	0.974	1	0.955	1	1	0.98	1
5	0.906	0.934	0.906	0.945	0.925	0.925	0.887	0.906
6	0.939	0.876	0.857	0.864	0.959	0.959	0.857	0.959
7	0.904	0.908	0.904	0.925	0.865	0.885	0.923	0.904
8	0.854	0.872	0.833	0.909	0.812	0.896	0.896	0.833
9	0.878	0.936	0.829	0.853	0.829	0.780	0.854	0.854

continued on next page

Table A.2.: Two-Class 70/30 Test Results

Phone	**SVM1**	**SVM1 CV**	**SVM2**	**SVM2 CV**	**RF1**	**RF2**	**ML**	**VC**
10	0.908	0.966	0.908	0.860	0.877	0.892	0.908	0.892
11	0.985	0.927	0.954	0.922	0.969	0.954	0.969	0.954
12	0.667	0.778	0.593	0.860	0.778	0.630	0.630	0.630
13	0.909	0.910	0.886	0.901	0.909	0.909	0.955	0.955
14	0.920	0.930	0.920	0.848	0.840	0.900	0.960	0.940
15	0.906	0.782	0.938	0.845	0.969	0.938	0.938	0.938
16	0.758	0.935	0.758	0.971	0.879	0.848	0.788	0.788
17	0.964	0.891	0.964	0.913	0.964	0.964	0.929	0.964
18	0.971	0.938	0.986	0.964	0.986	0.986	0.986	0.986
19	0.907	0.840	0.930	0.902	0.860	0.907	0.860	0.860
20	0.971	0.975	0.971	0.981	0.957	0.957	0.957	0.957
21	0.966	0.926	0.966	0.927	0.966	0.966	0.966	0.966
22	0.932	0.926	0.881	0.947	0.898	0.915	0.915	0.932
23	0.958	0.863	0.875	0.873	0.896	0.958	0.896	0.917
24	0.981	0.952	0.963	0.913	0.944	0.963	0.944	0.963
25	0.972	0.926	0.944	0.922	0.972	0.972	0.944	0.972
26	0.932	0.951	0.932	0.930	0.955	0.955	0.955	0.932
27	0.895	0.896	0.789	0.874	0.895	0.947	0.947	0.895
28	0.905	0.890	0.905	0.897	0.889	0.954	0.952	0.937
29	0.917	0.955	0.938	0.874	0.854	0.875	0.917	0.896
30	0.892	0.941	0.919	0.817	0.973	0.946	0.973	0.973
31	0.933	0.856	0.889	0.834	0.911	0.867	0.911	0.911
32	0.870	0.929	0.870	0.928	0.852	0.889	0.907	0.889
33	0.952	0.969	0.952	0.947	0.952	0.929	0.976	0.952
34	1	0.863	0.896	0.858	0.979	0.979	0.979	0.979
35	0.938	0.963	0.951	0.927	0.963	0.975	0.975	0.975
36	0.977	0.902	0.955	0.932	0.977	0.977	0.955	1
Average	0.921	0.912	0.904	0.906	0.917	0.922	0.923	0.923
Attack	**Sessions**							
4	0	0.948	0	0.955	0.200	0.200	0	0

continued on next page

Table A.2.: Two-Class 70/30 Test Results

Phone	**SVM1**	**SVM1 CV**	**SVM2**	**SVM2 CV**	**RF1**	**RF2**	**ML**	**VC**
15	0.400	0.754	0	0.845	0.800	0.400	0	0.200
16	0	0.934	0	0.971	0.200	0.200	0	0
27	0	0.828	0	0.874	0.200	0	0	0
32	0	0.937	0	0.928	0.400	0.600	0.200	0.400
Average	0.080	0.880	0	0.915	0.360	0.280	0.040	0.12

Table A.3.: Two-Class 30/70 Test Results

Phone	**SVM1**	**SVM1 CV**	**SVM2**	**SVM2 CV**	**RF1**	**RF2**	**ML**	**VC**
1	0.927	0.921	0.949	0.871	0.972	0.972	0.966	0.972
3	0.909	0.924	0.909	0.935	0.813	0.850	0.941	0.925
4	0.974	0.980	0.974	0.967	0.966	0.948	0.974	0.983
5	0.877	0.865	0.934	0.867	0.918	0.885	0.926	0.910
6	0.895	0.960	0.877	0.963	0.904	0.904	0.930	0.895
7	0.909	0.942	0.909	0.908	0.785	0.826	0.884	0.876
8	0.865	0.808	0.865	0.86	0.838	0.874	0.914	0.874
9	0.863	0.776	0.863	0.893	0.8	0.821	0.874	0.863
10	0.900	0.861	0.9	0.957	0.767	0.867	0.953	0.913
11	0.934	1	0.934	1	0.921	0.907	0.960	0.960
12	0.571	0.926	0.571	0.758	0.587	0.635	0.667	0.651
13	0.941	0.719	0.882	0.797	0.824	0.873	0.843	0.931
14	0.922	0.917	0.922	0.935	0.848	0.887	0.913	0.904
15	0.595	0.775	0.757	0.800	0.784	0.757	0.784	0.811
16	0.883	0.908	0.883	0.975	0.857	0.792	0.896	0.883
17	0.899	0.8	0.899	0.85	0.884	0.884	0.915	0.891
18	0.957	0.928	0.957	0.959	0.957	0.969	0.963	0.975
19	0.861	0.762	0.891	0.758	0.891	0.911	0.891	0.881
20	0.963	0.942	0.963	0.936	0.932	0.938	0.963	0.951
21	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
22	0.927	1	0.927	0.963	0.883	0.883	0.934	0.942

continued on next page

Table A.3.: Two-Class 30/70 Test Results

Phone	**SVM1**	**SVM1 CV**	**SVM2**	**SVM2 CV**	**RF1**	**RF2**	**ML**	**VC**
23	0.864	0.853	0.864	0.855	0.864	0.864	0.918	0.873
24	0.960	0.963	0.944	0.986	0.960	0.960	0.937	0.960
25	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
26	0.883	0.953	0.883	0.905	0.816	0.883	0.903	0.864
27	0.932	0.727	0.864	0.758	0.898	0.909	0.977	0.966
28	0.878	0.870	0.878	0.881	0.844	0.850	0.939	0.891
29	0.848	0.893	0.777	0.858	0.857	0.848	0.848	0.839
30	0.844	0.862	0.884	0.922	0.919	0.895	0.953	0.930
31	0.838	0.701	0.848	0.702	0.790	0.810	0.886	0.857
32	0.905	0.944	0.905	0.960	0.881	0.905	0.944	0.921
33	0.959	0.804	0.928	0.830	0.969	0.979	0.928	0.969
34	0.873	0.852	0.873	0.983	0.918	0.927	0.891	0.936
35	0.968	0.963	0.937	0.872	0.968	0.968	0.958	0.968
36	0.951	0.839	0.942	0.875	0.961	0.981	0.922	0.971
Average	0.887	0.877	0.888	0.889	0.872	0.884	0.912	0.907
Attack	**Sessions**							
4	0.200	0.959	0.200	0.967	0.200	0.400	0.200	0.200
15	0.200	0.719	0.400	0.800	0.400	0.400	0.400	0.400
16	0	0.937	0	0.975	0	0	0	0
27	0.200	0.811	0	0.758	0.200	0.200	0.200	0.200
32	0	0.963	0	0.960	0.400	0.200	0	0
Average	0.120	0.8778	0.120	0.892	0.240	0.240	0.160	0.160

B. ONE-CLASS MODELS - AVERAGE RESULTS

Table B.1.: One-Class 80/20 Test Results

Phone	Linear	RBF	Polynomial	Sigmoid
1	1	0.5	0	1
3	0.52	0.52	0.040	0.84
4	0	0.330	0.330	1
5	0.600	0.400	0.200	1
6	1	0	0.200	0.800
7	0.333	0.444	0.222	0.555
8	0.556	0.667	0	0.444
9	0.500	0.400	0.200	0.600
10	0.727	0.455	0.272	0.636
11	0	0.500	0	0.500
12	0.375	0.875	0.125	0.750
13	0.571	0.571	0	0.857
14	0.800	0.300	0.300	1
15	0.833	0.333	0.333	0.833
16	0.375	0.500	0.500	0.500
17	0.667	0.333	0.167	0.833
18	0	0.500	0	0.250
19	0.800	0.200	0.400	0.600
20	0.750	0.500	0	0.750
21	1	0	1	0

continued on next page

Table B.1.: One-Class 80/20 Test Results

Phone	**Linear**	**RBF**	**Polynomial**	**Sigmoid**
22	0.750	0.250	0.750	0
23	0.400	0	0.200	0.600
24	1	0	0	1
25	0	0	1	0
26	0.500	0.333	0.167	0.500
27	0.250	0	0	1
28	0.333	0.500	0.167	0.833
29	0.800	0.400	0.200	0.800
30	0.250	0.500	0	0.750
31	0.333	0.444	0.222	0.778
32	0.500	0.375	0.125	0.875
33	0.500	0	0	0.500
34	0	0.333	0	0.667
35	0.500	0	0	1
36	0	0.500	0	0.500
Average	0.501	0.342	0.203	0.673
Attack	**Sessions**			
15	0.800	0.200	0.800	0.600
16	0.600	0.600	0	0.800
27	0.200	0.400	0.200	0.800
32	0.400	0.200	0	1
Average	0.500	0.350	0.250	0.800

Table B.2.: One-Class 70/30 Test Results

Phone	**Linear**	**RBF**	**Polynomial**	**Sigmoid**
1	0.333	0.333	0.333	0.667
3	0.579	0.579	0.026	0.763
4	0.250	0.250	0.250	0.75
5	0	0.571	0	0.571
6	0.571	0.143	0.1413	0.714
7	0.500	0.500	0.143	0.429
8	0.462	0.462	0.154	0.538
9	0.800	0.200	0	0.800
10	0.529	0.412	0.412	0.647
11	1	0.500	0	0.333
12	0.330	0.583	0.167	0.750
13	0.400	0.600	0	0.500
14	0.267	0.333	0.133	0.933
15	0.778	0.333	0	0.889
16	0.333	0.333	0.583	0.333
17	0.111	0.444	0.222	0.556
18	0.833	0.167	0.500	0.333
19	0.143	0.286	0.286	0.571
20	0.333	0.333	0.167	0.500
21	0.500	0	0	1
22	0.667	0.167	0.667	0.167
23	0.429	0.143	0.143	0.571
24	0.667	0	0	1
25	0	0	0.500	0.500
26	0.444	0.222	0.111	0.667

continued on next page

Table B.2.: One-Class 70/30 Test Results

Phone	**Linear**	**RBF**	**Polynomial**	**Sigmoid**
27	0.500	0.167	0	0.833
28	0	0.500	0.250	0.750
29	0.143	0.571	0	0.857
30	0.167	0.500	0	0.667
31	0.615	0.385	0.077	0.923
32	0.417	0.250	0.167	0.833
33	0.500	0	0	0.500
34	0.600	0	0.200	0.200
35	0.333	0	0	1
36	0.333	0	0.333	1
Average	0.425	0.293	0.170	0.658
Attack	**Sessions**			
4	0.600	0	0	1
15	0.800	0.200	0.200	0.400
16	0.400	0.200	0	0.600
27	0.800	0.200	0.400	0.800
32	0.200	0.200	0	1
Average	0.560	0.160	0.120	0.760

Table B.3.: One-Class 30/70 Test Results

Phone	**Linear**	**RBF**	**Polynomial**	**Sigmoid**
1	0	0	0.286	0.286
3	0.804	0.517	0.170	0.896

continued on next page

Table B.3.: One-Class 30/70 Test Results

Phone	Linear	RBF	Polynomial	Sigmoid
4	0	0	0.250	0.750
5	0.500	0.313	0.063	0.813
6	0.237	0.267	0.200	0.867
7	0.516	0.161	0.161	0.709
8	0.552	0.310	0.067	0.966
9	0.686	0.114	0.057	0.743
10	0.513	0.282	0.179	0.641
11	0.357	0	0.071	0.643
12	0.615	0.269	0.154	0.538
13	0.391	0.261	0	0.870
14	0.333	0.091	0.091	0.879
15	0.667	0.286	0.286	0.905
16	0.571	0	0.107	0.857
17	0.600	0	0.100	0.950
18	0.692	0	0.462	0.462
19	0.333	0.067	0.267	0.733
20	0.583	0	0.083	0.917
22	0.500	0.071	0.071	0.786
23	0.313	0.063	0.313	0.500
24	0	0	0.400	0.400
26	0.600	0.150	0.100	0.750
27	0.538	0	0.231	0.769
28	0.579	0	0.158	0.842
29	0.063	0.063	0.313	0.625
30	0.357	0	0.286	0.643

continued on next page

Table B.3.: One-Class 30/70 Test Results

Phone	**Linear**	**RBF**	**Polynomial**	**Sigmoid**
31	0.552	0.172	0.172	0.586
32	0.536	0.107	0.214	0.500
33	0	0	0	0
34	0.200	0	0.100	0.900
35	0	0	0.500	0.500
36	0.714	0	0.429	0.857
Average	0.421	0.108	0.192	0.699
Attack	**Sessions**			
4	0.400	0	0.200	1
15	0.600	0	0.400	0.800
16	0.600	0	0.200	0.600
27	0.200	0.200	0.200	0.800
32	0.800	0	0.400	1
Average	0.520	0.040	0.280	0.840

C. ROC CURVES 80/20 TRAINING TEST SPLIT

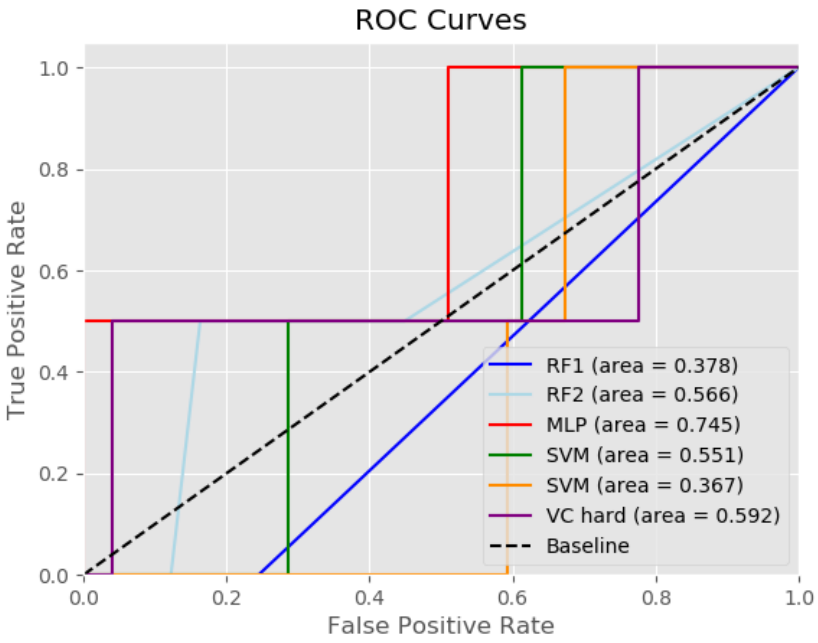

Fig. C.1. Phone 1: 80/20 Training/Test split

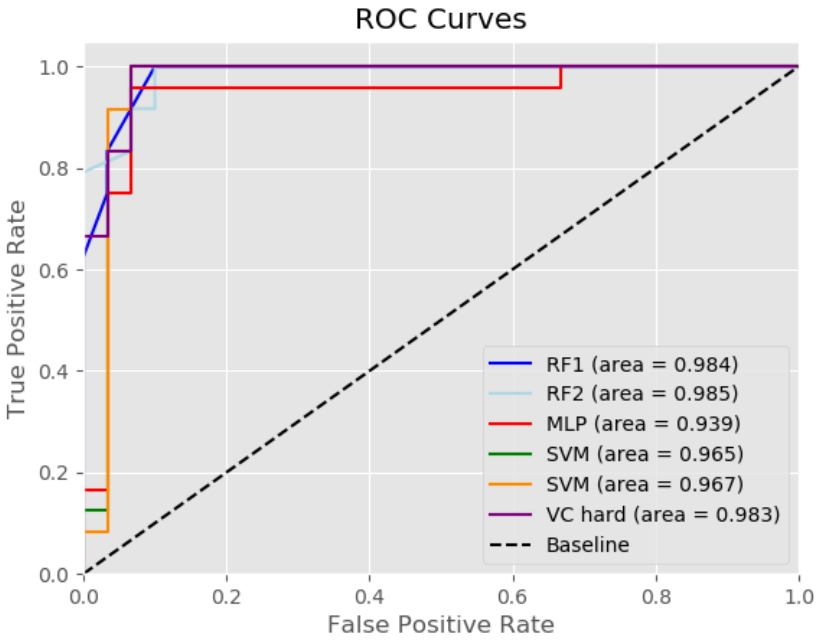

Fig. C.2. Phone 3: 80/20 Training/Test split

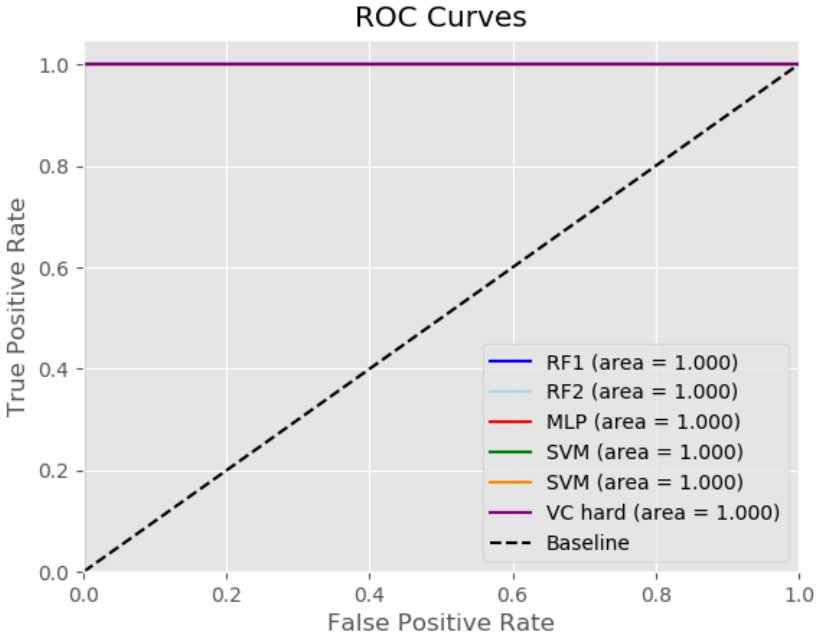

Fig. C.3. Phone 4: 80/20 Training/Test split

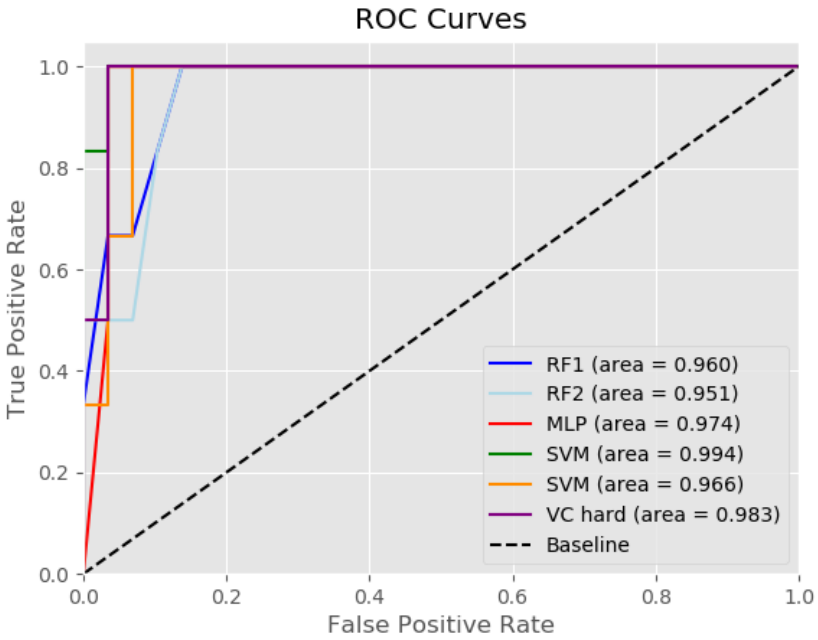

Fig. C.4. Phone 5: 80/20 Training/Test split

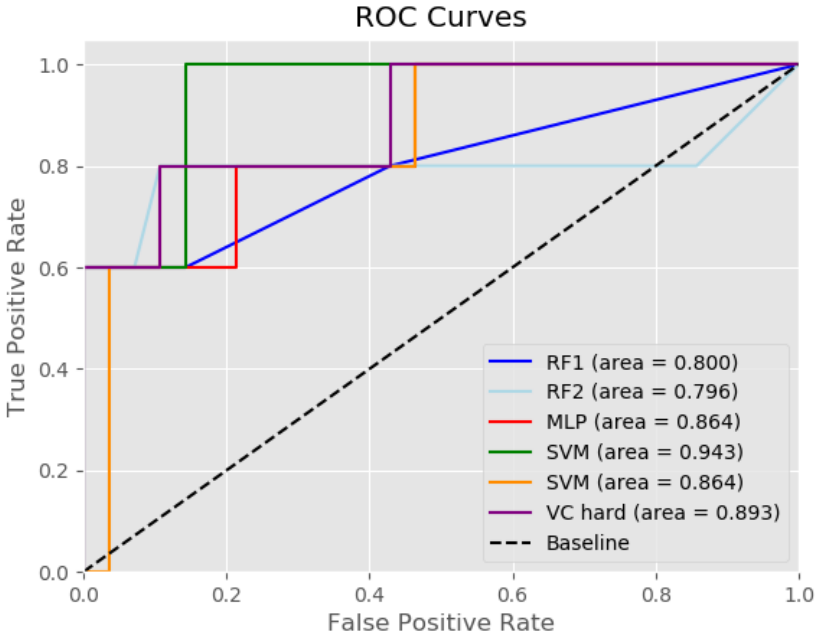

Fig. C.5. Phone 6: 80/20 Training/Test split

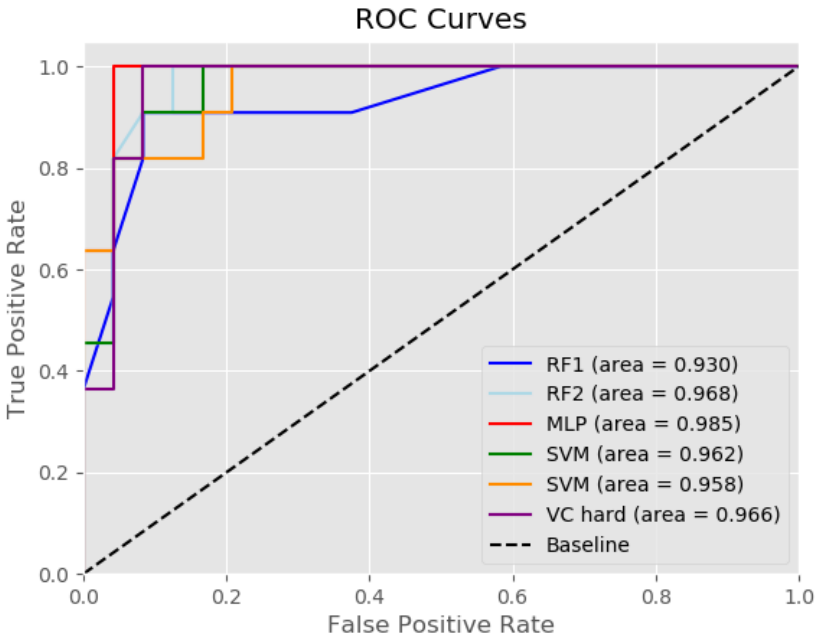

Fig. C.6. Phone 7: 80/20 Training/Test split

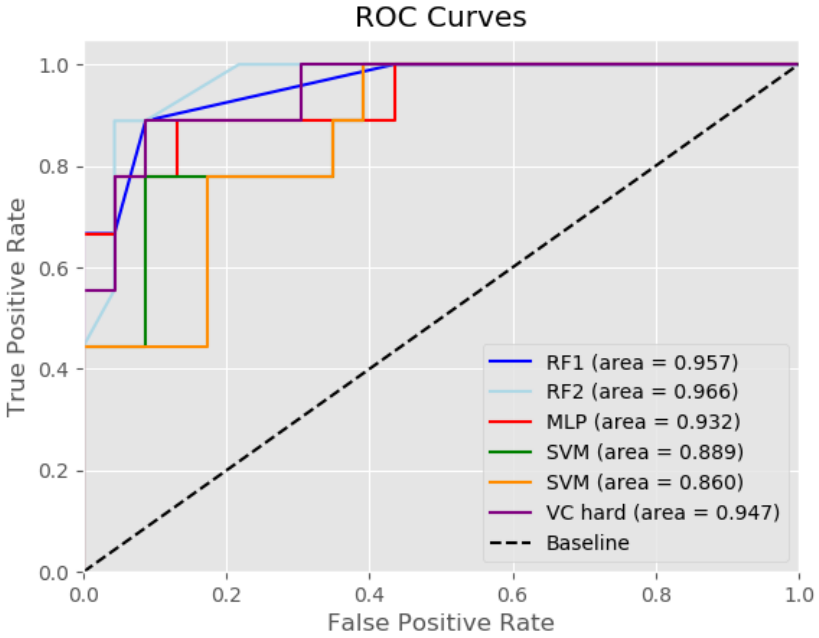

Fig. C.7. Phone 8: 80/20 Training/Test split

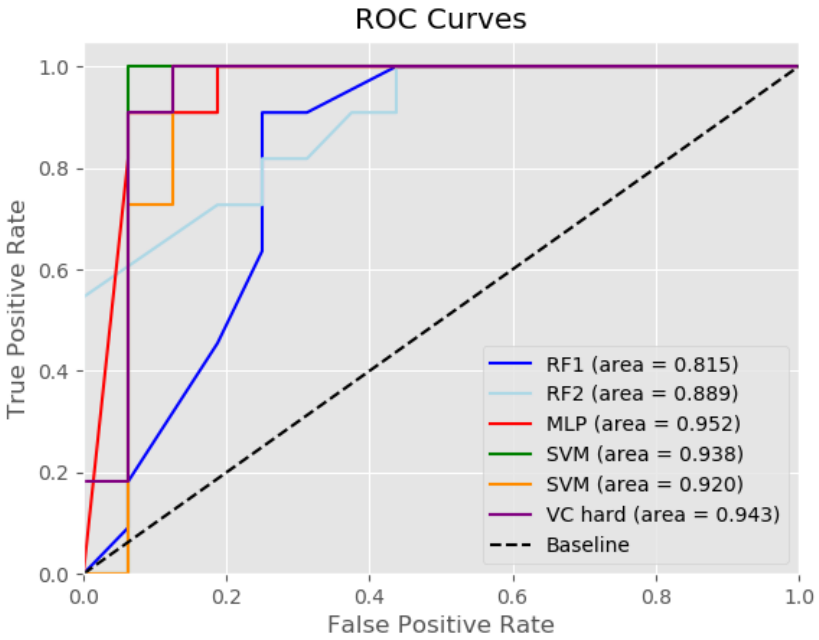

Fig. C.8. Phone 9: 80/20 Training/Test split

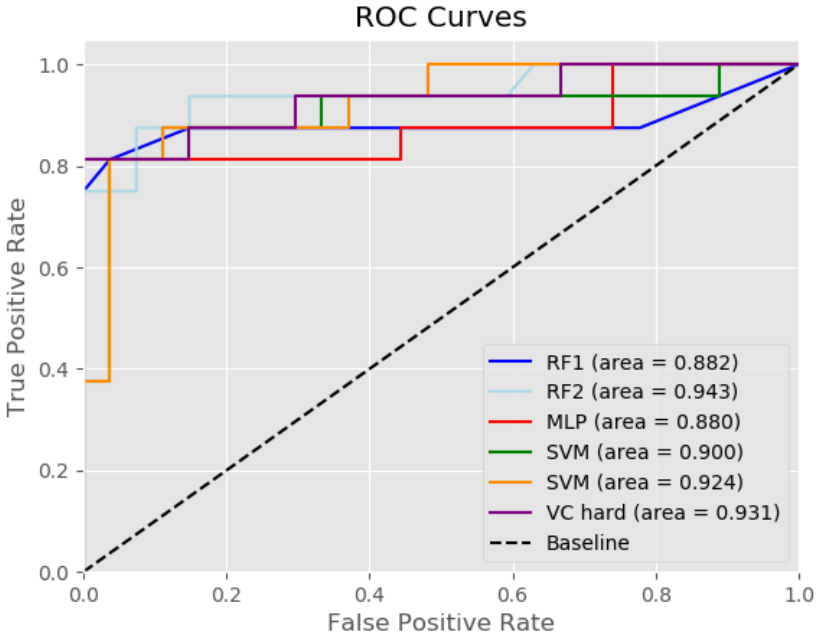

Fig. C.9. Phone 10: 80/20 Training/Test split

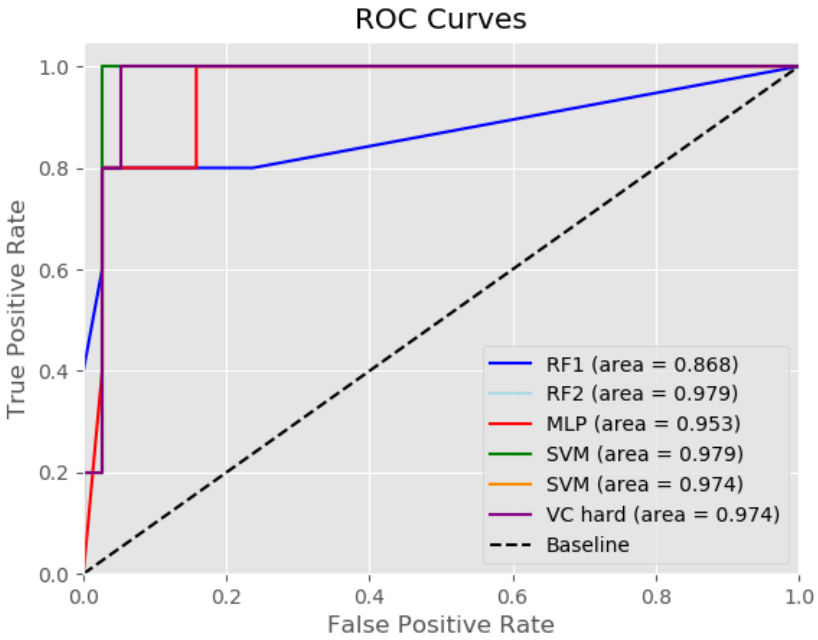

Fig. C.10. Phone 11: 80/20 Training/Test split

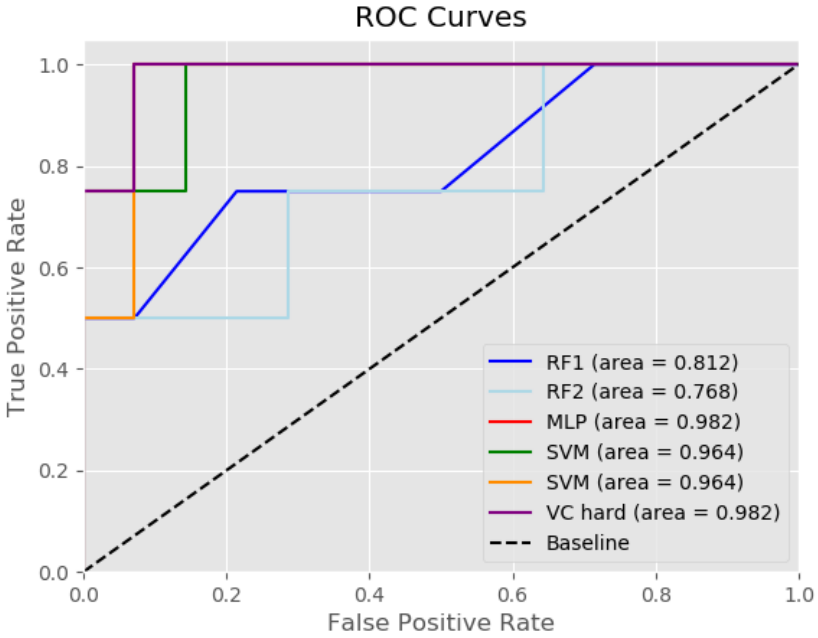

Fig. C.11. Phone 12: 80/20 Training/Test split

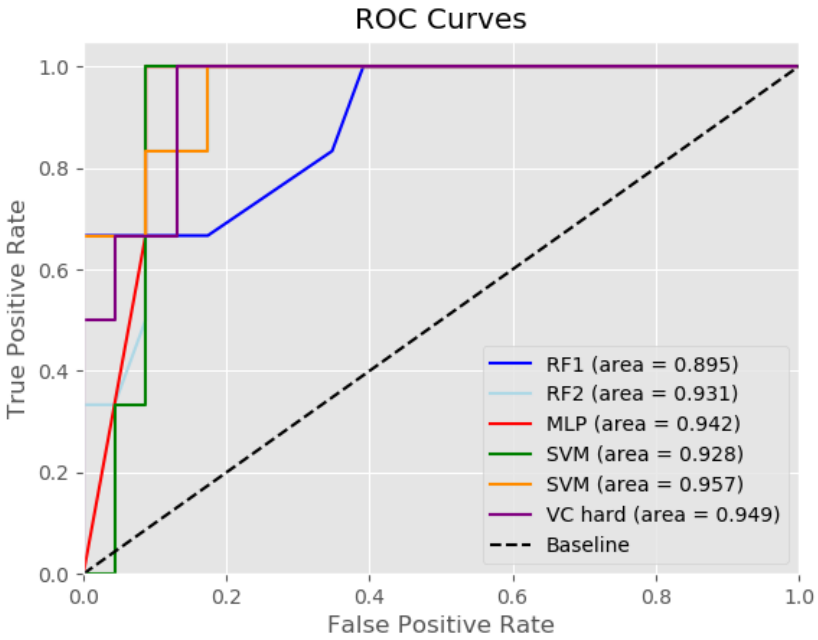

Fig. C.12. Phone 13: 80/20 Training/Test split

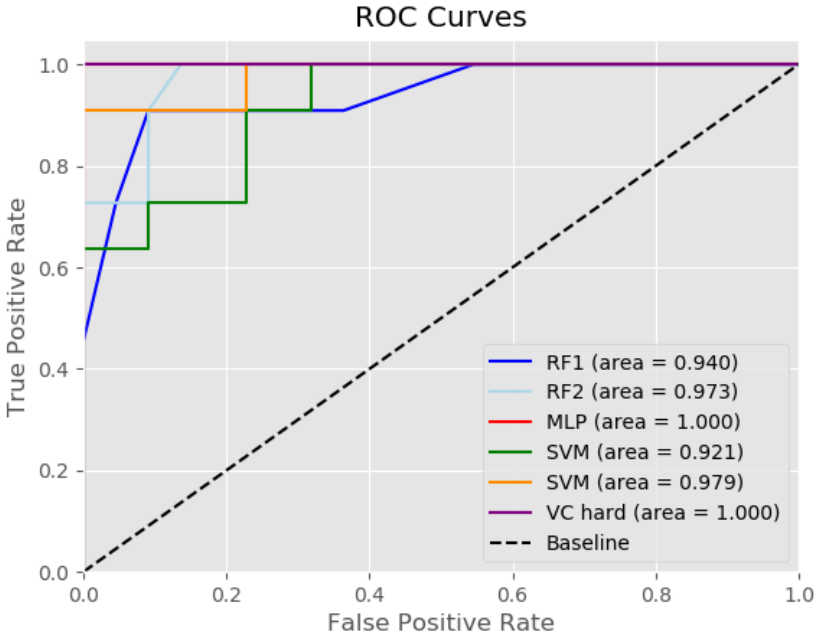

Fig. C.13. Phone 14: 80/20 Training/Test split

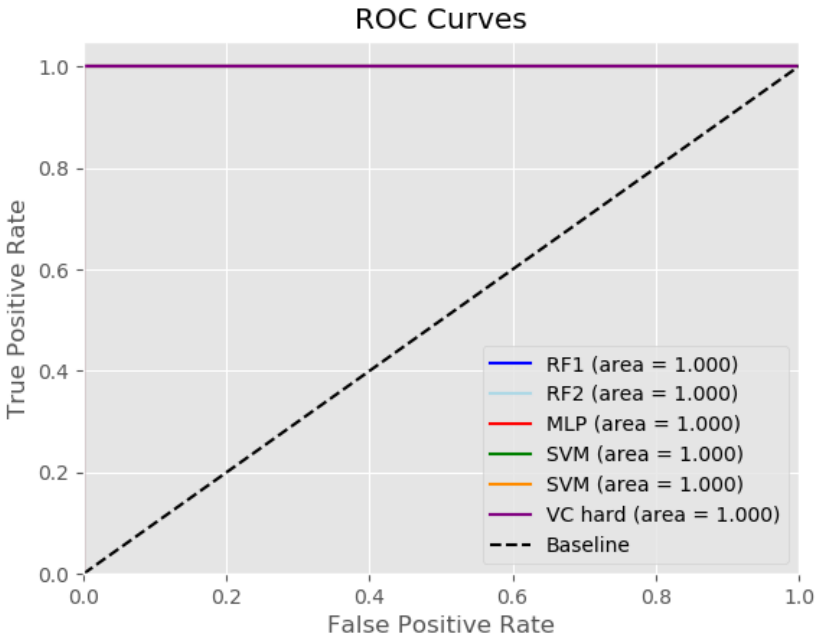

Fig. C.14. Phone 15: 80/20 Training/Test split

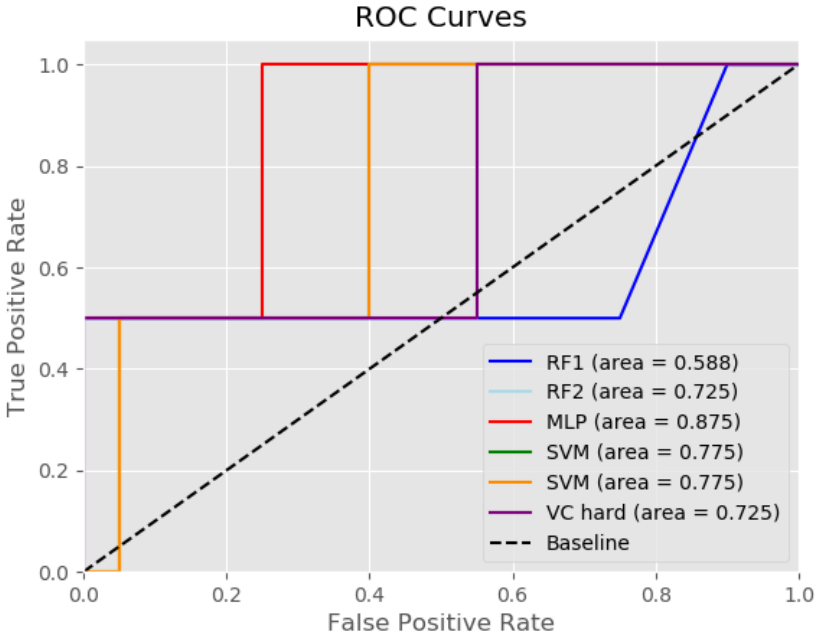

Fig. C.15. Phone 16: 80/20 Training/Test split

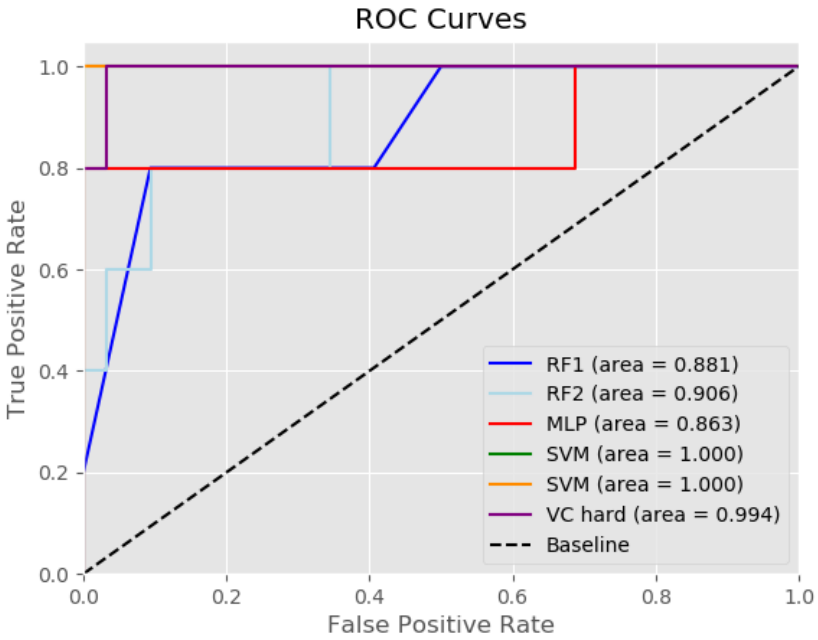

Fig. C.16. Phone 17: 80/20 Training/Test split

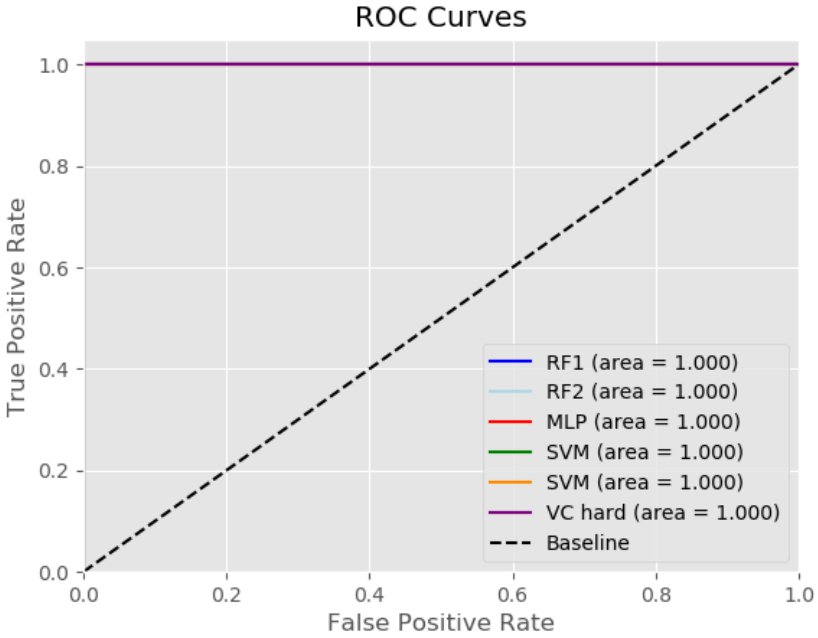

Fig. C.17. Phone 18: 80/20 Training/Test split

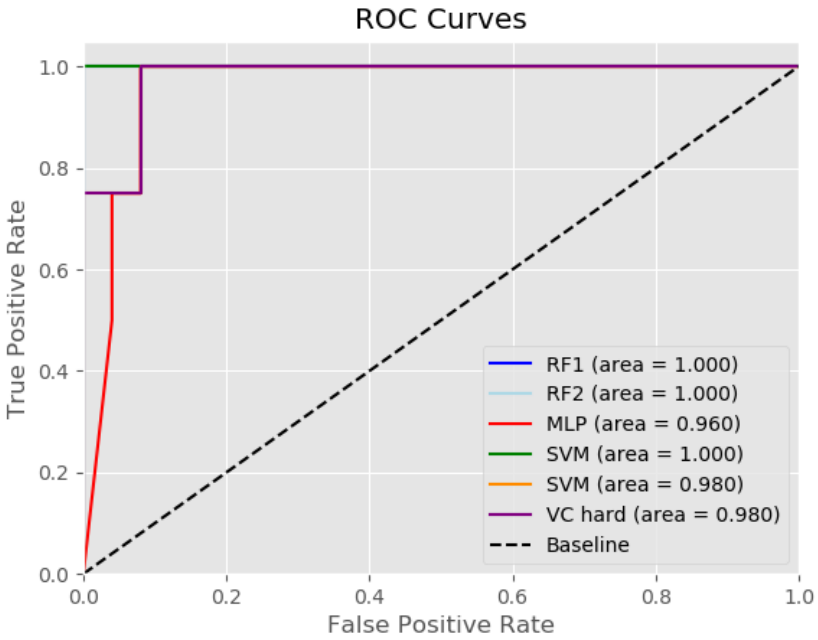

Fig. C.18. Phone 19: 80/20 Training/Test split

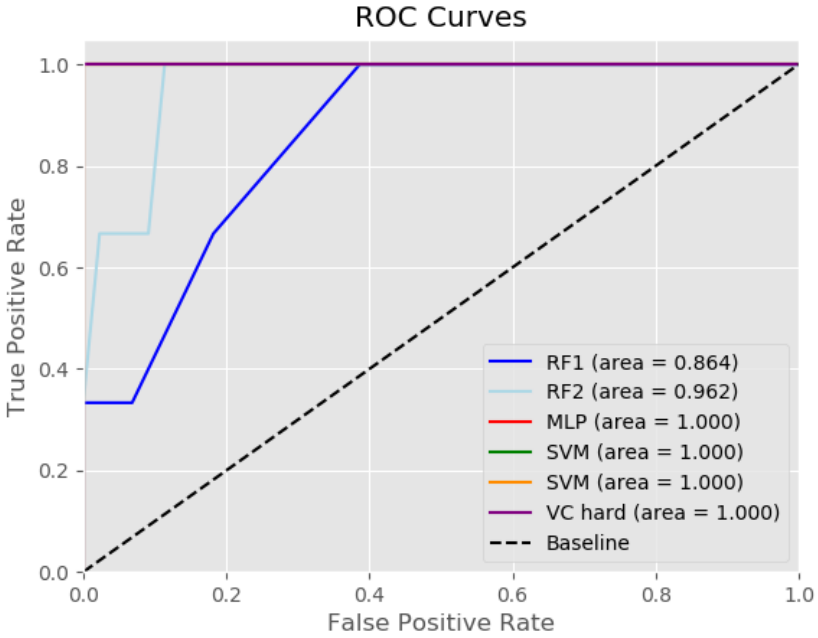

Fig. C.19. Phone 20: 80/20 Training/Test split

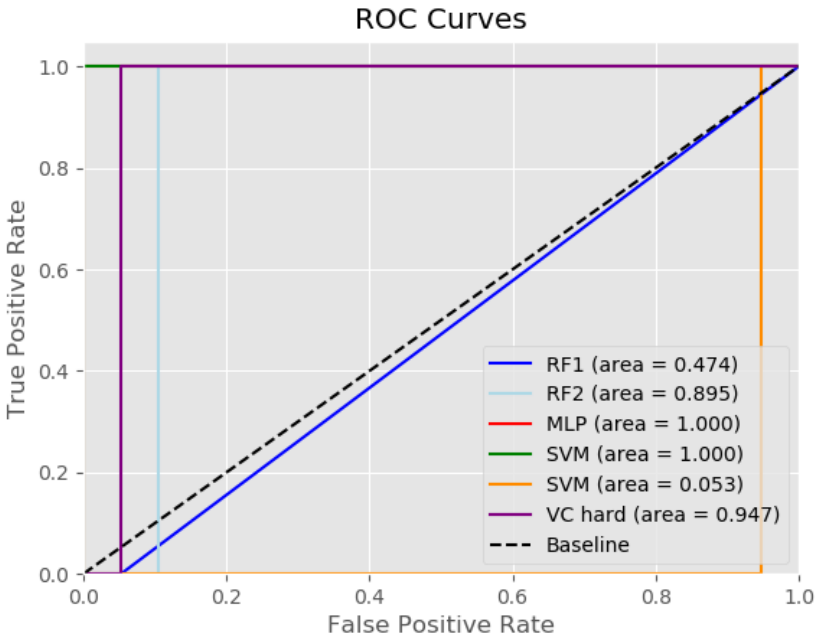

Fig. C.20. Phone 21: 80/20 Training/Test split

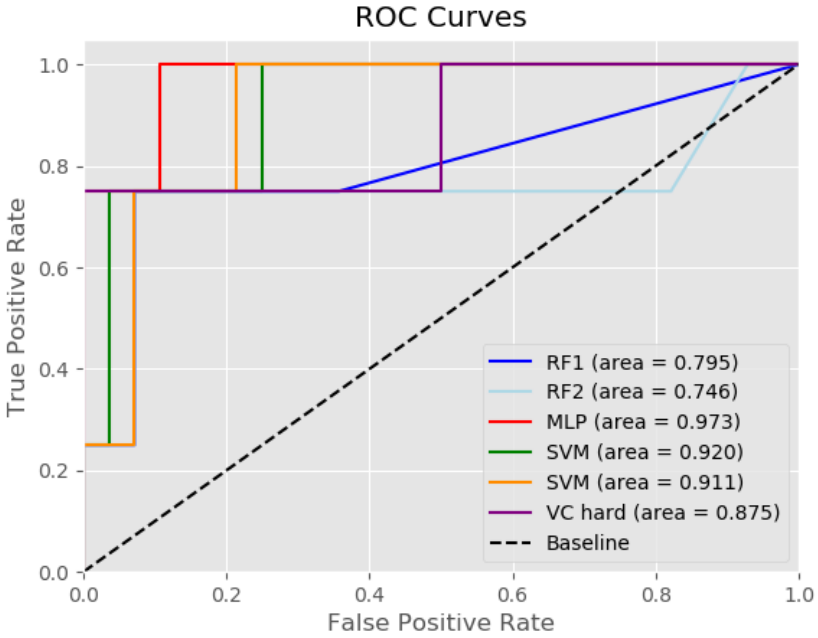

Fig. C.21. Phone 23: 80/20 Training/Test split

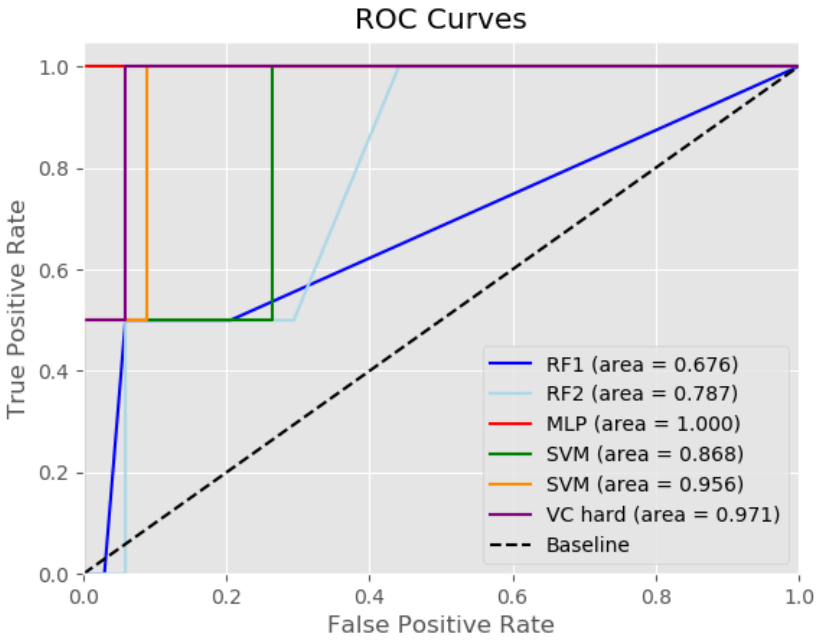

Fig. C.22. Phone 24: 80/20 Training/Test split

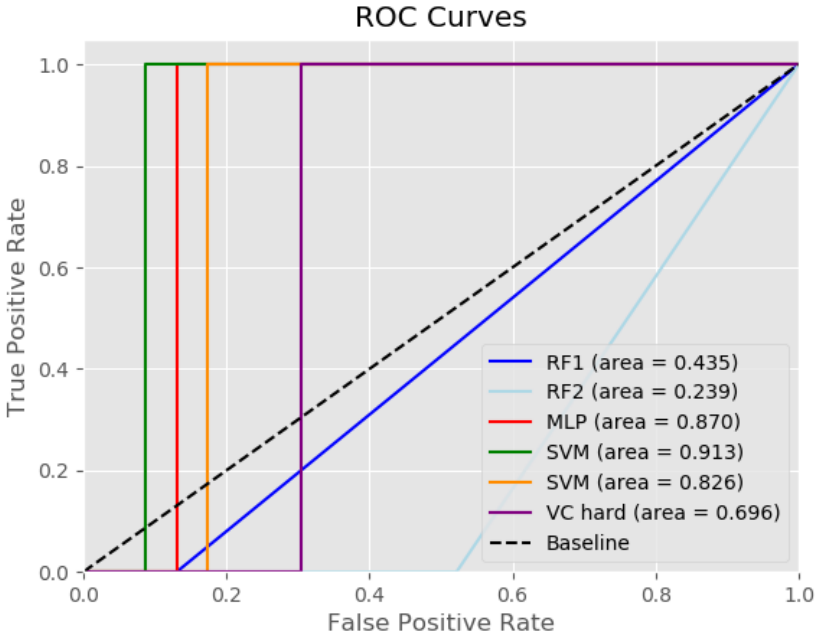

Fig. C.23. Phone 25: 80/20 Training/Test split

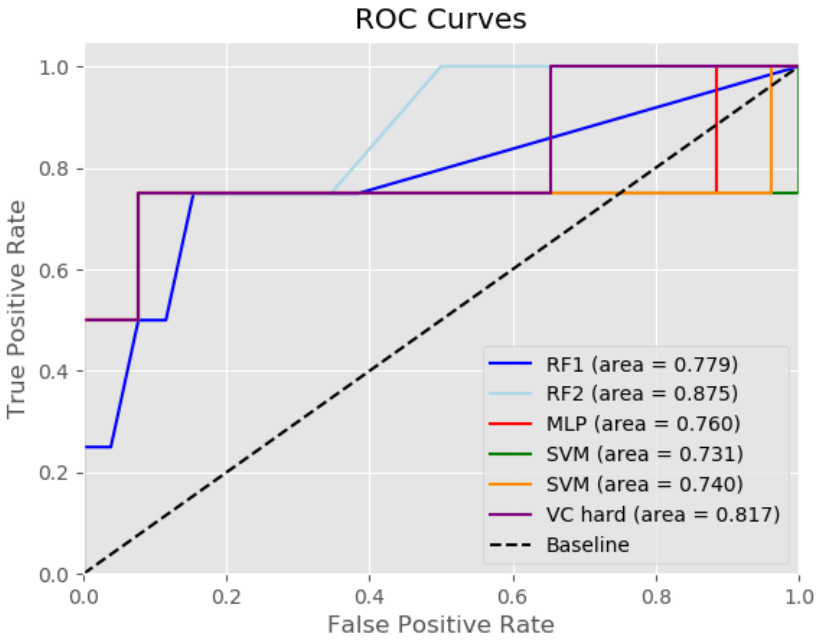

Fig. C.24. Phone 26: 80/20 Training/Test split

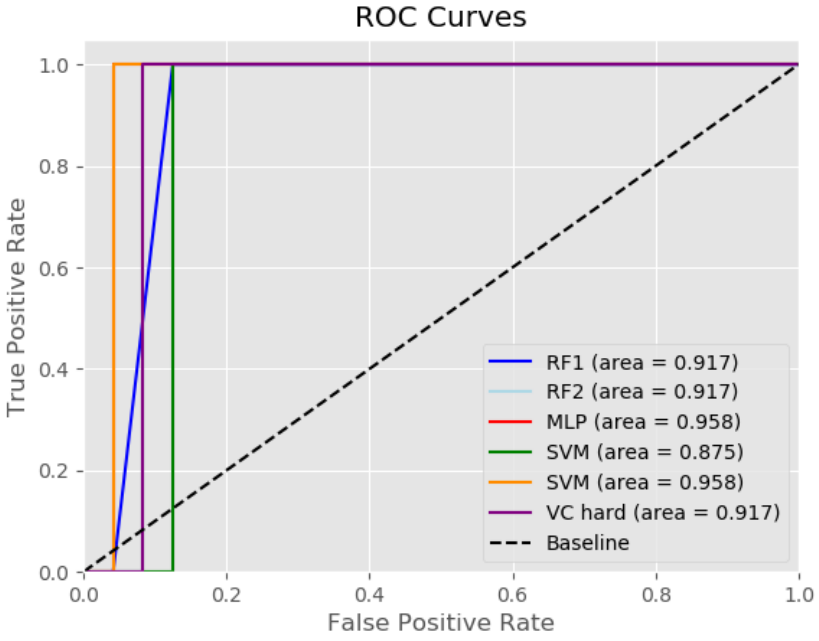

Fig. C.25. Phone 27: 80/20 Training/Test split

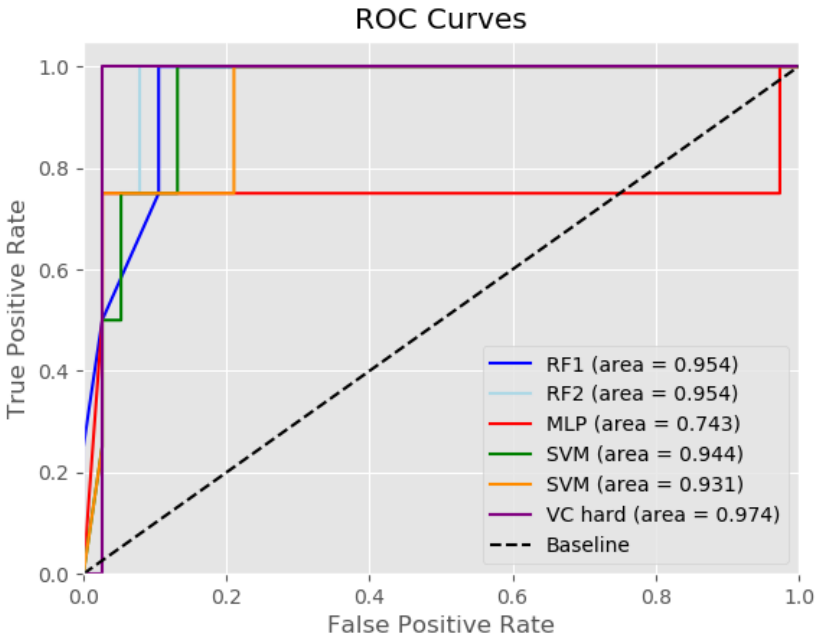

Fig. C.26. Phone 28: 80/20 Training/Test split

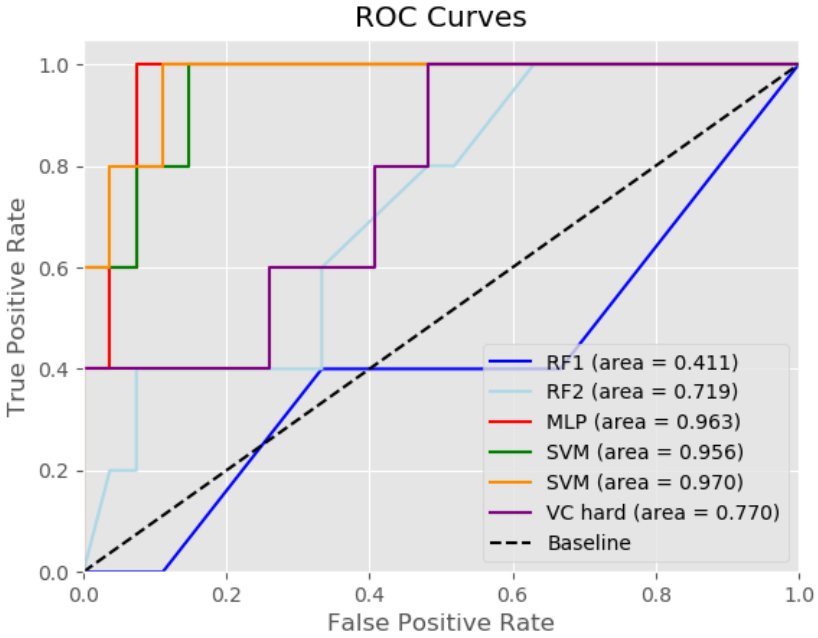

Fig. C.27. Phone 29: 80/20 Training/Test split

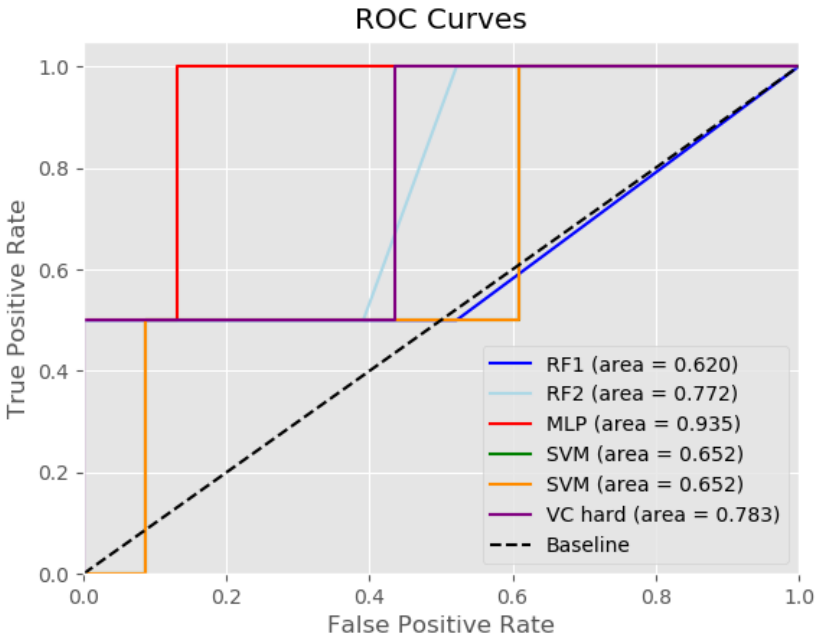

Fig. C.28. Phone 30: 80/20 Training/Test split

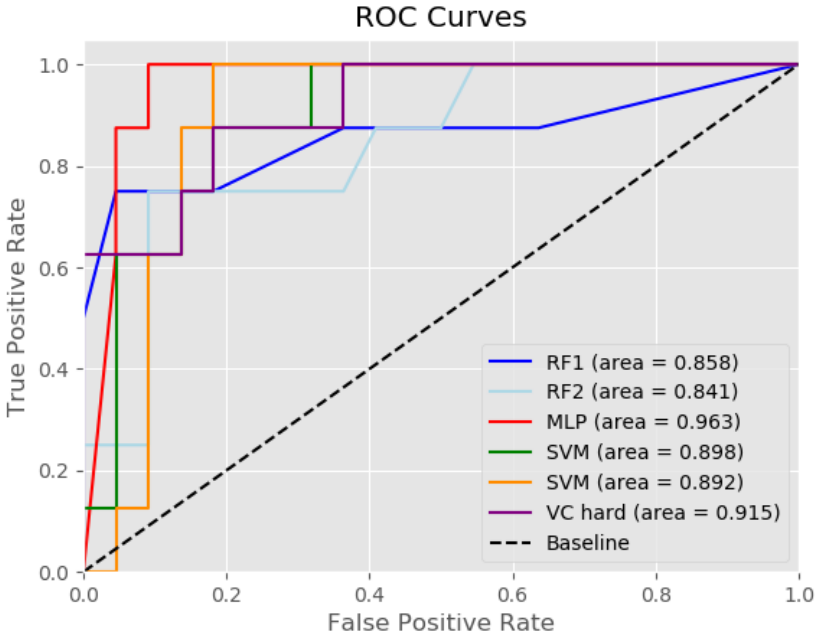

Fig. C.29. Phone 31: 80/20 Training/Test split

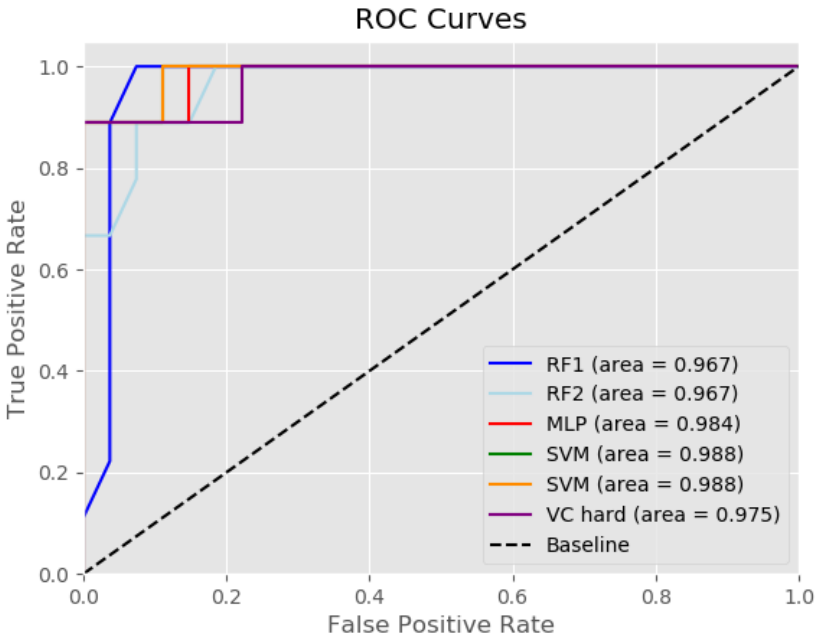

Fig. C.30. Phone 32: 80/20 Training/Test split

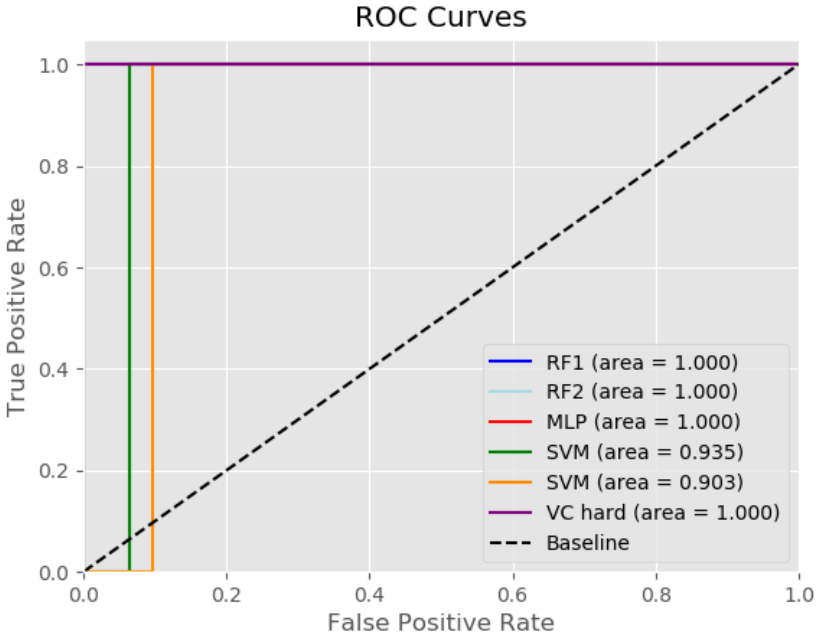

Fig. C.31. Phone 34: 80/20 Training/Test split

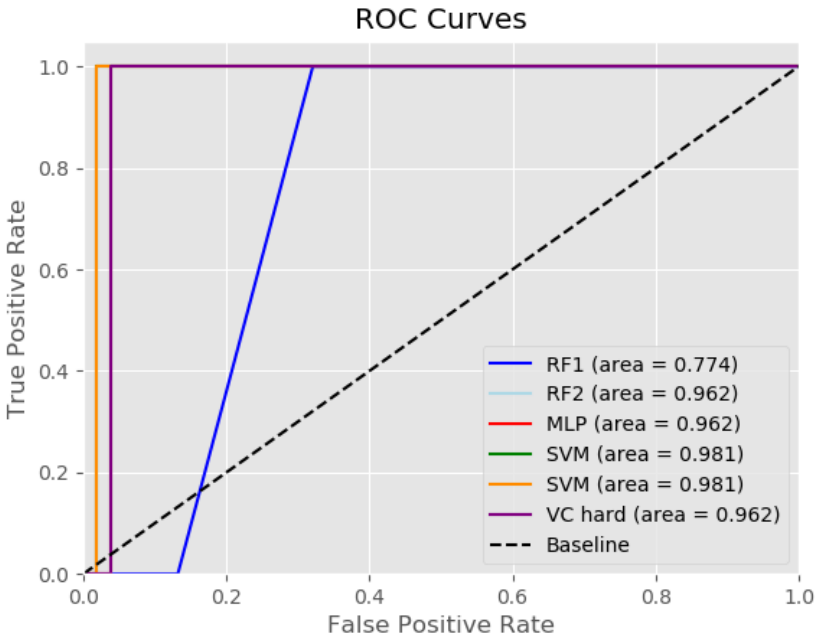

Fig. C.32. Phone 35: 80/20 Training/Test split

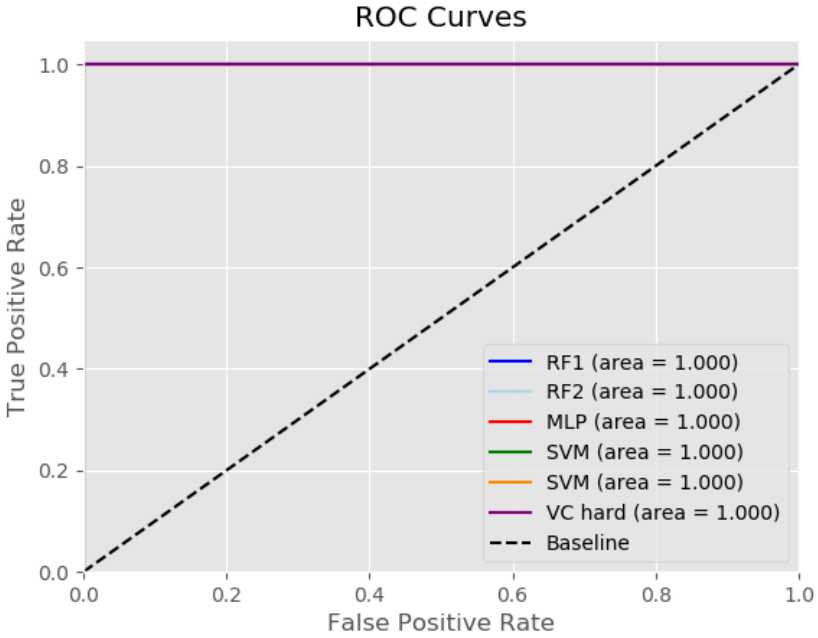

Fig. C.33. Phone 36: 80/20 Training/Test split

D. ROC CURVES 70/30 TRAINING TEST SPLIT

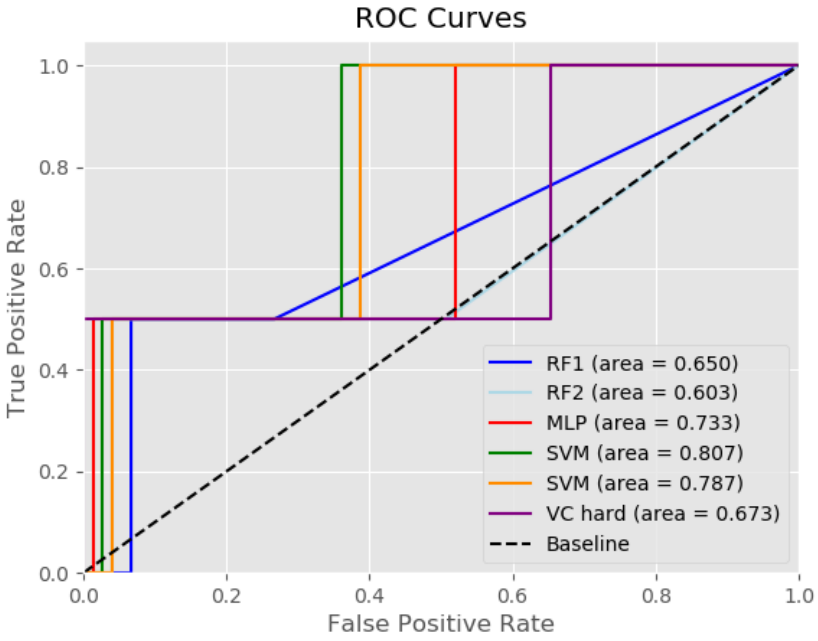

Fig. D.1. Phone 1: 70/30 Training/Test split

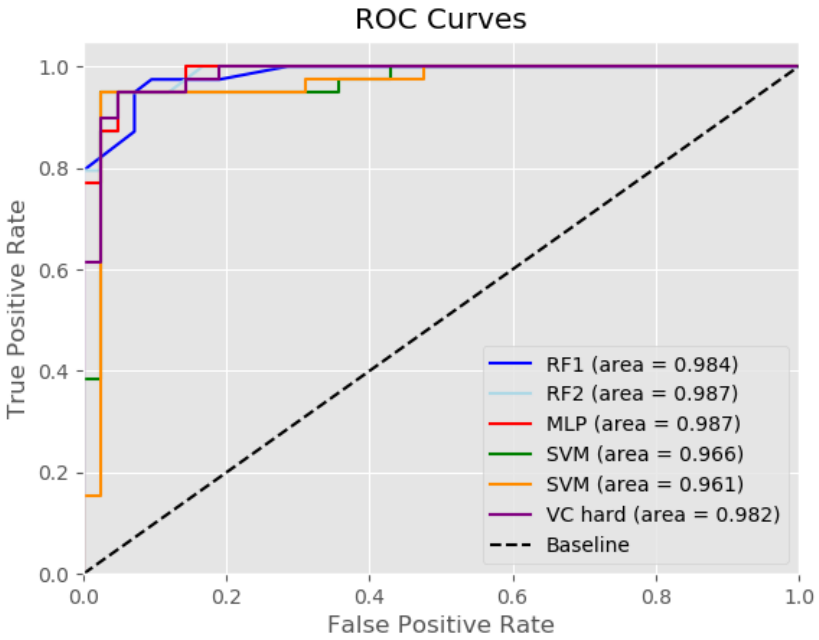

Fig. D.2. Phone 3: 70/30 Training/Test split

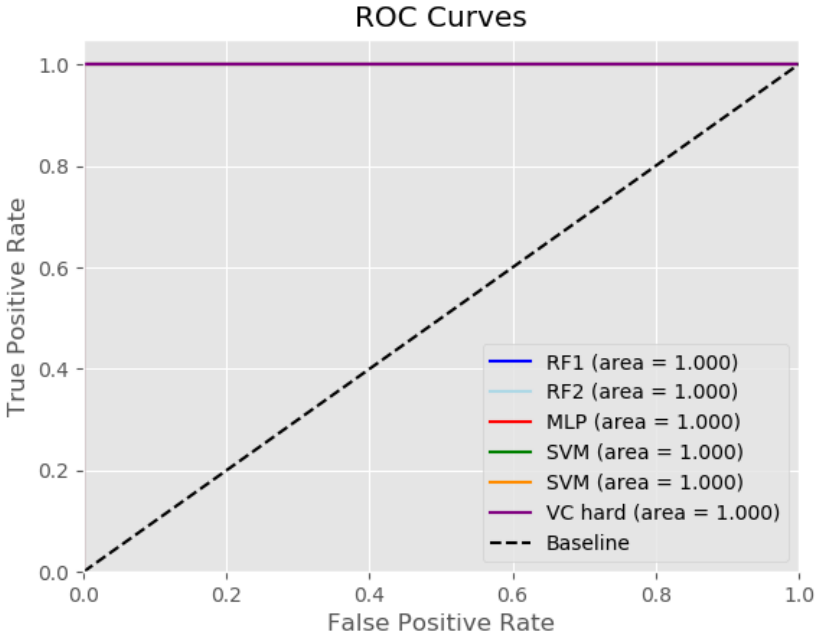

Fig. D.3. Phone 4: 70/30 Training/Test split

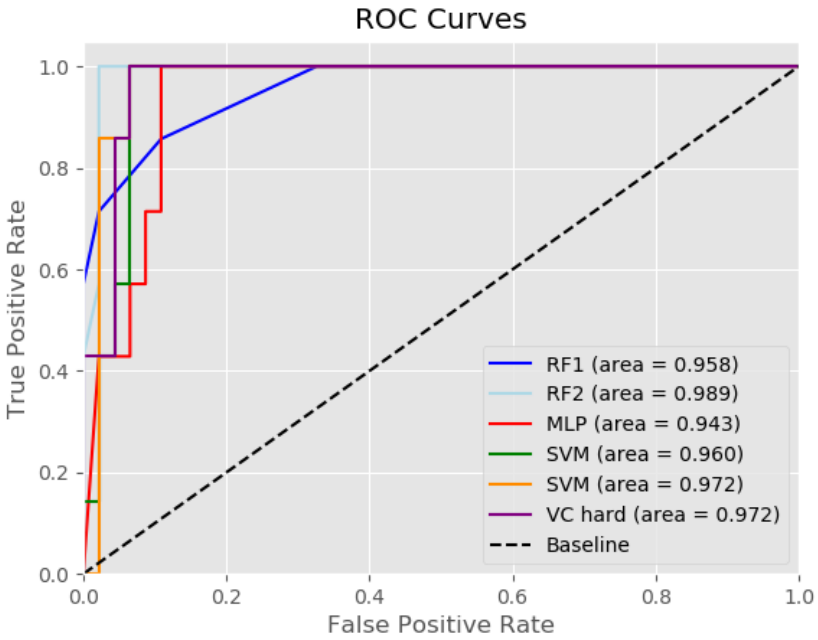

Fig. D.4. Phone 5: 70/30 Training/Test split

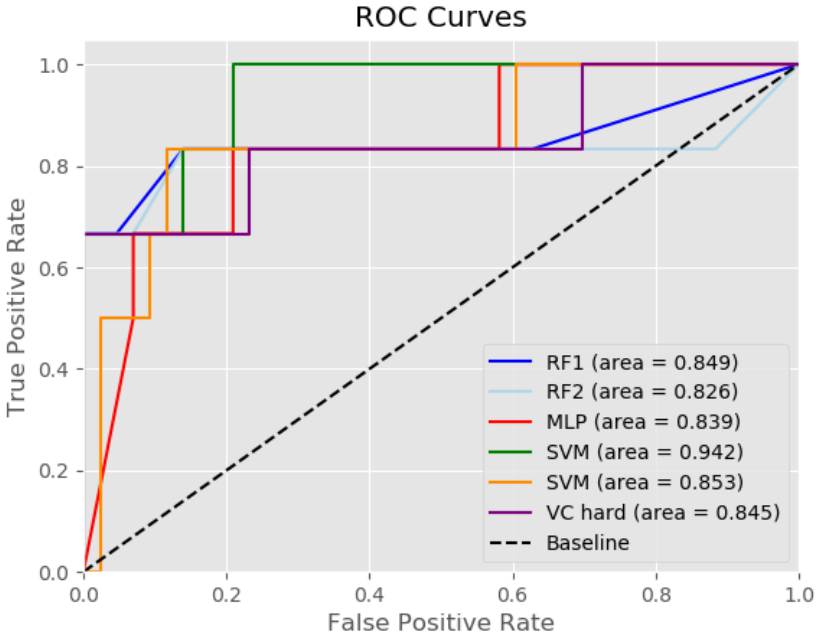

Fig. D.5. Phone 6: 70/30 Training/Test split

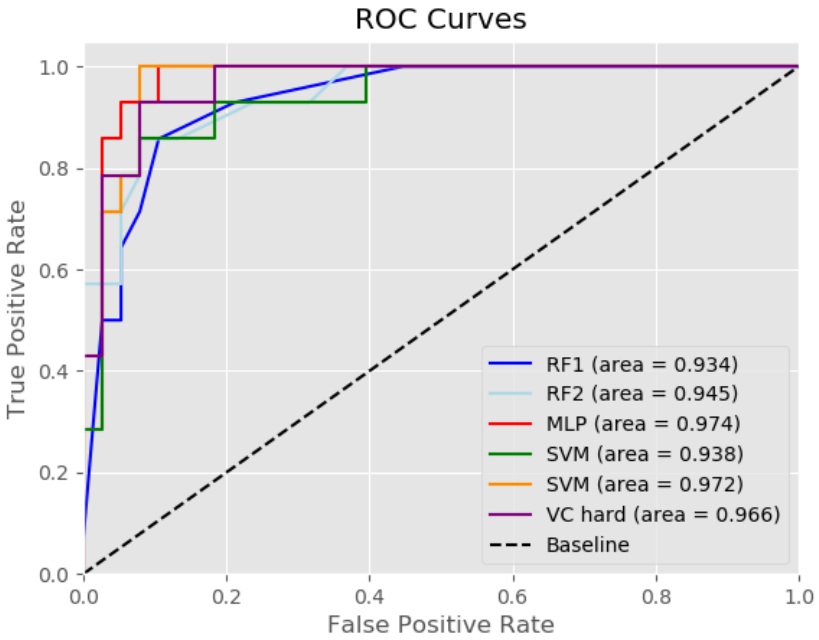

Fig. D.6. Phone 7: 70/30 Training/Test split

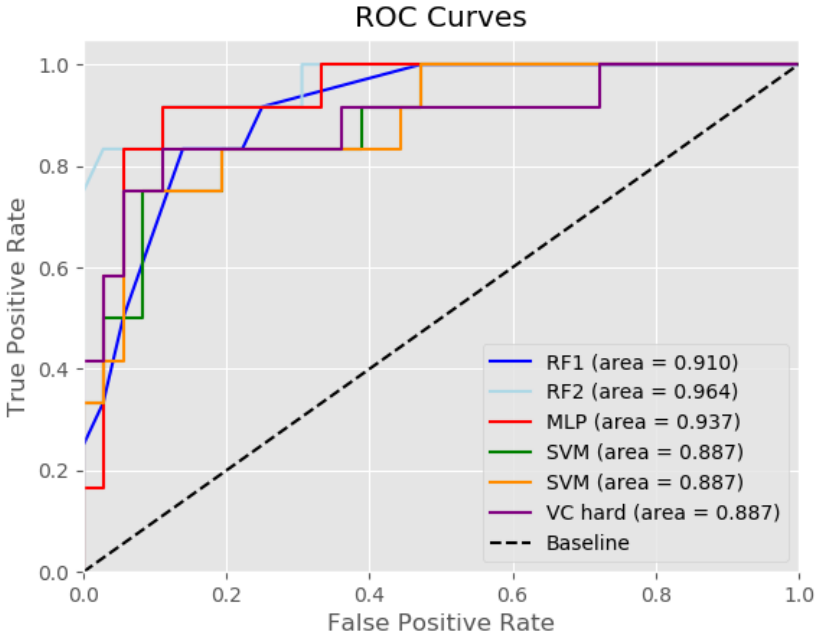

Fig. D.7. Phone 8: 70/30 Training/Test split

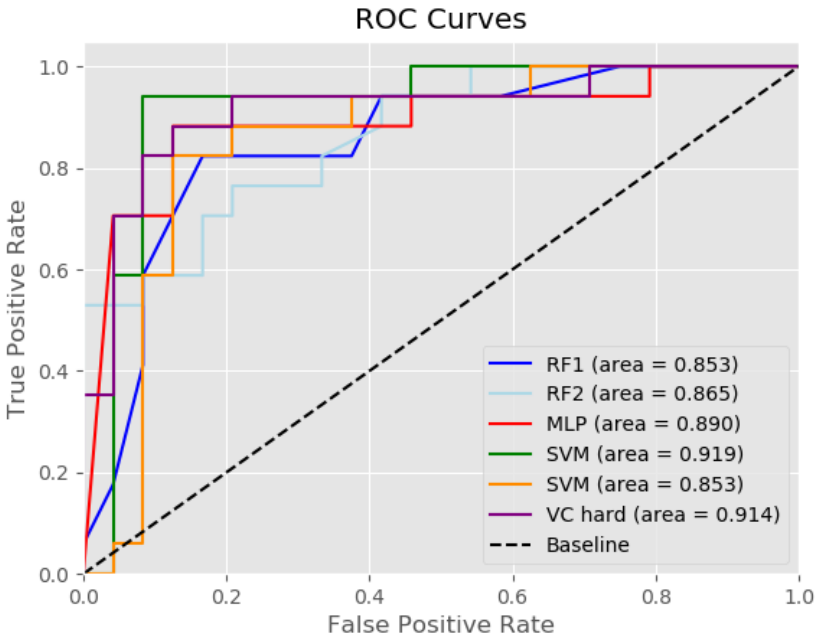

Fig. D.8. Phone 9: 70/30 Training/Test split

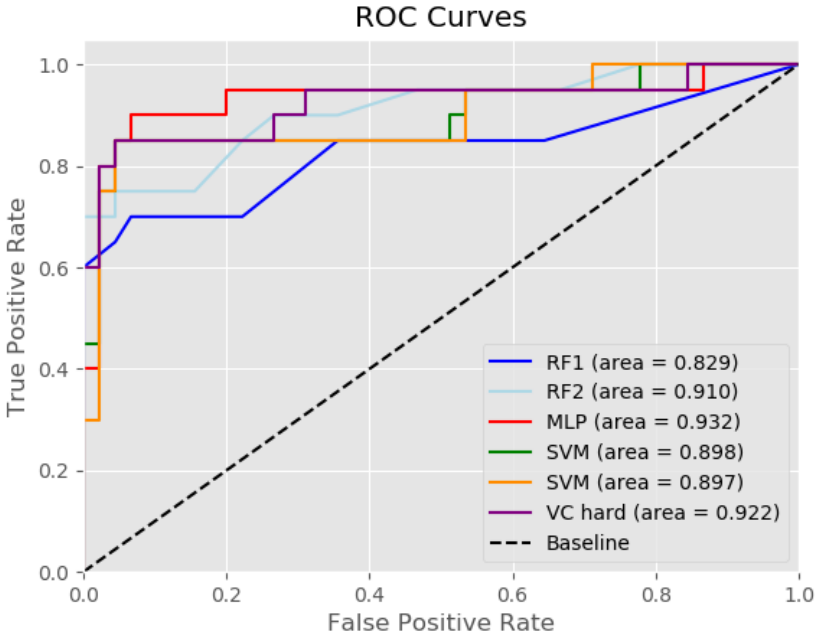

Fig. D.9. Phone 10: 70/30 Training/Test split

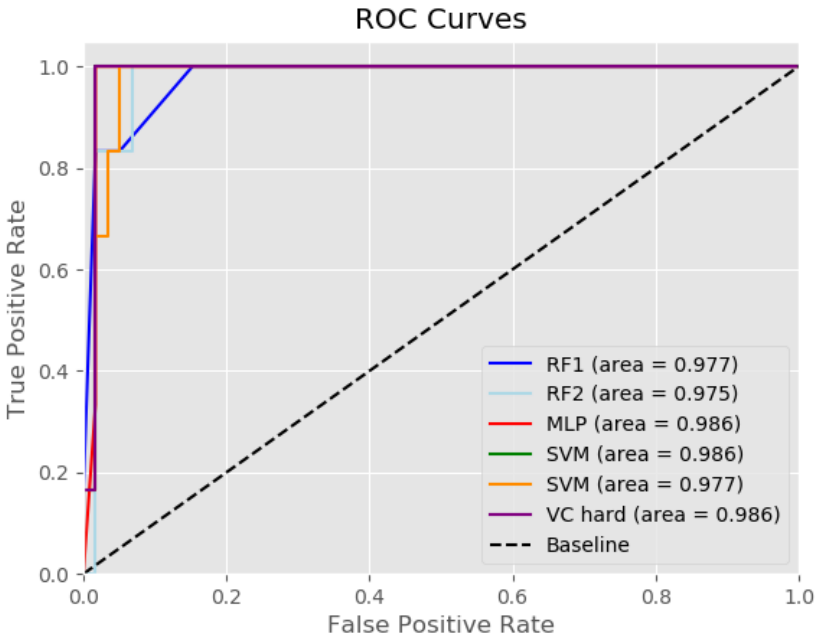

Fig. D.10. Phone 11: 70/30 Training/Test split

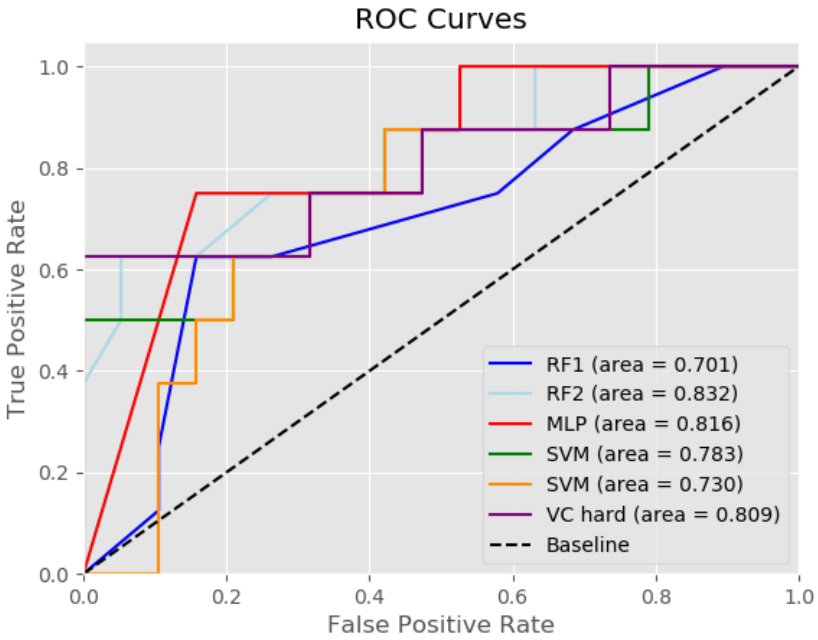

Fig. D.11. Phone 12: 70/30 Training/Test split

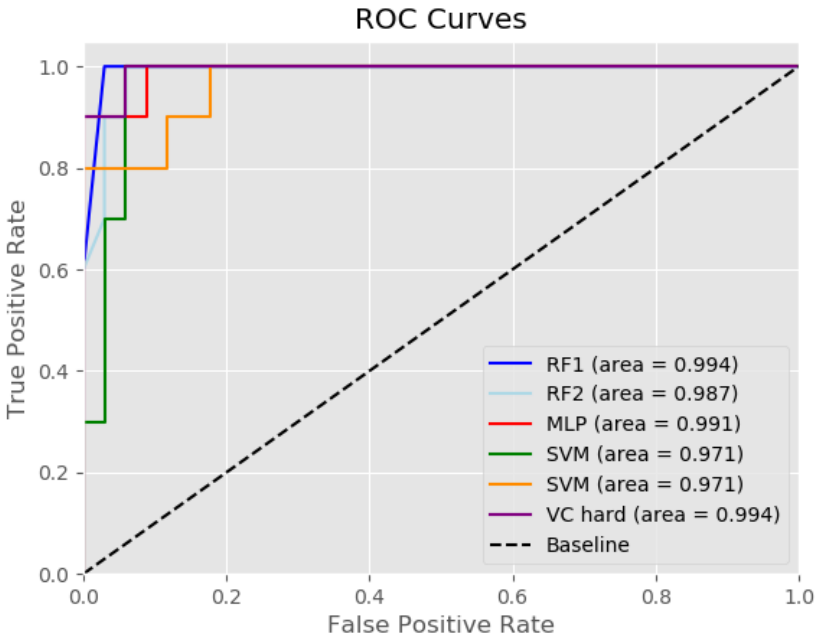

Fig. D.12. Phone 13: 70/30 Training/Test split

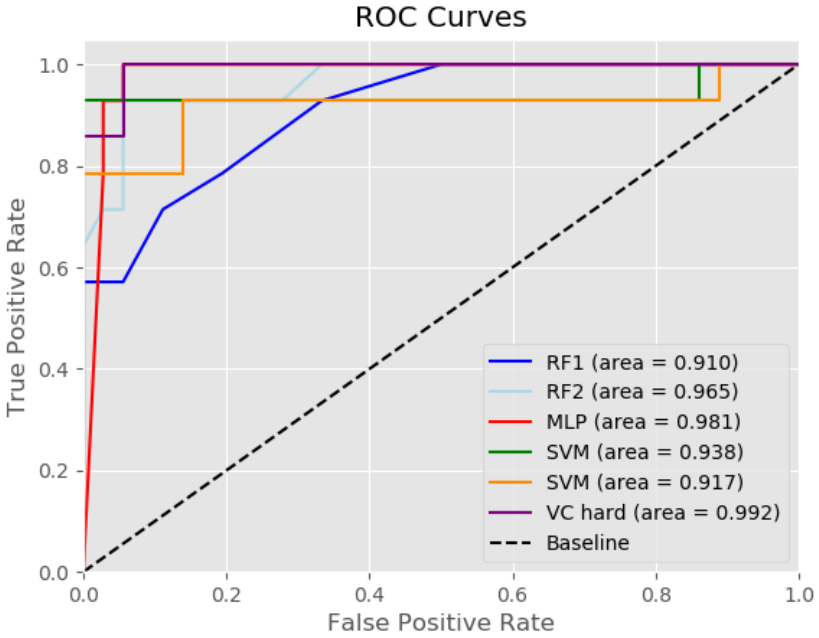

Fig. D.13. Phone 14: 70/30 Training/Test split

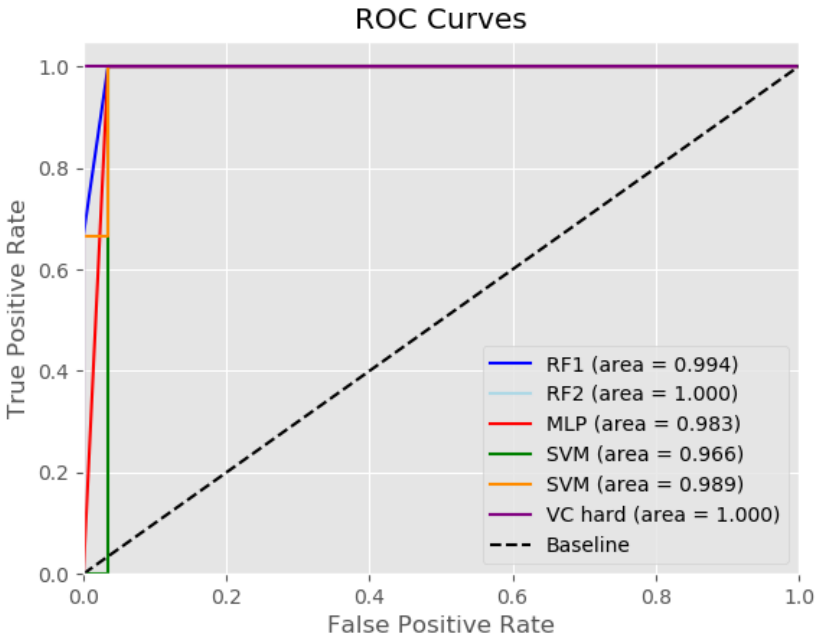

Fig. D.14. Phone 15: 70/30 Training/Test split

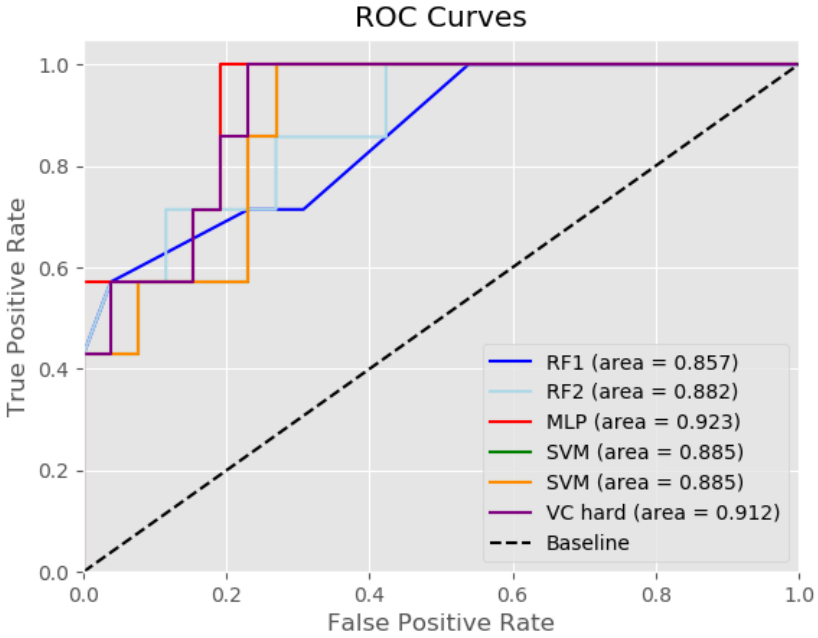

Fig. D.15. Phone 16: 70/30 Training/Test split

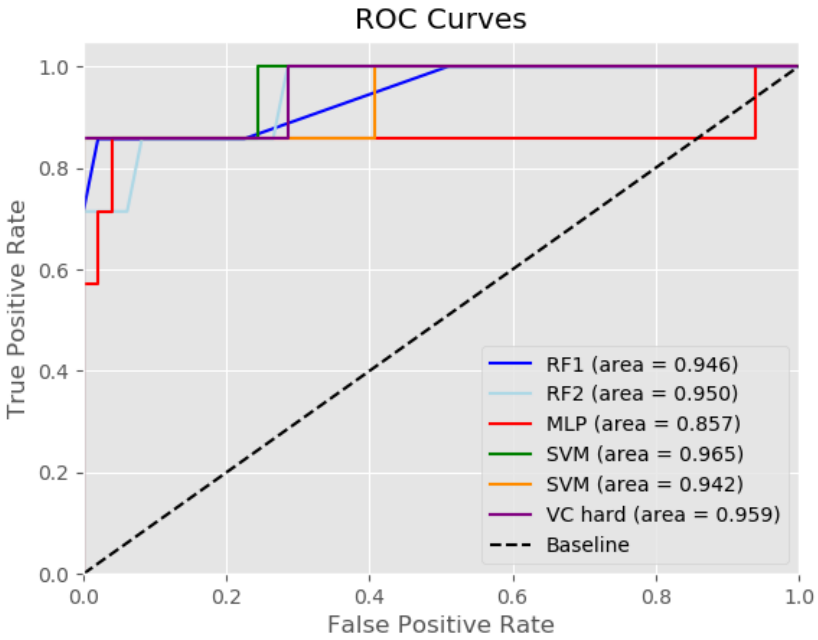

Fig. D.16. Phone 17: 70/30 Training/Test split

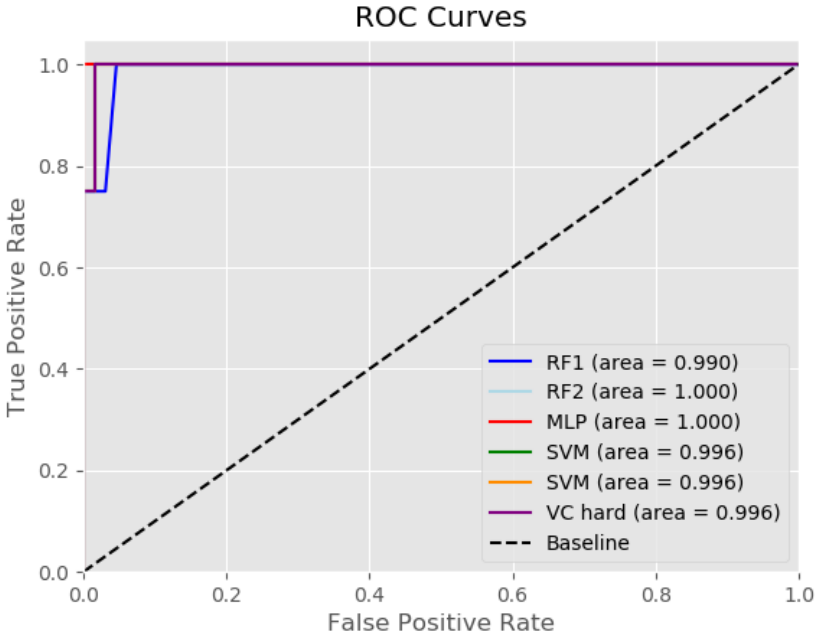

Fig. D.17. Phone 18: 70/30 Training/Test split

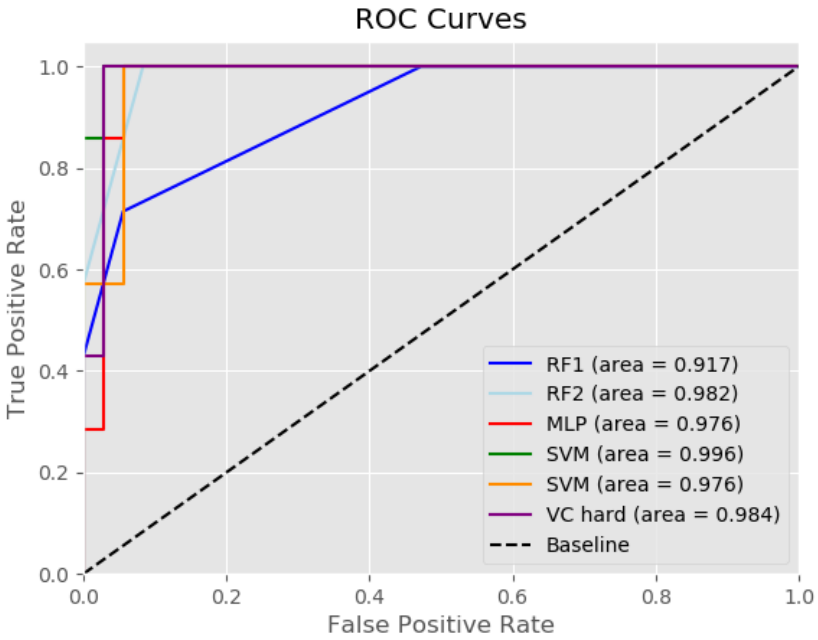

Fig. D.18. Phone 19: 70/30 Training/Test split

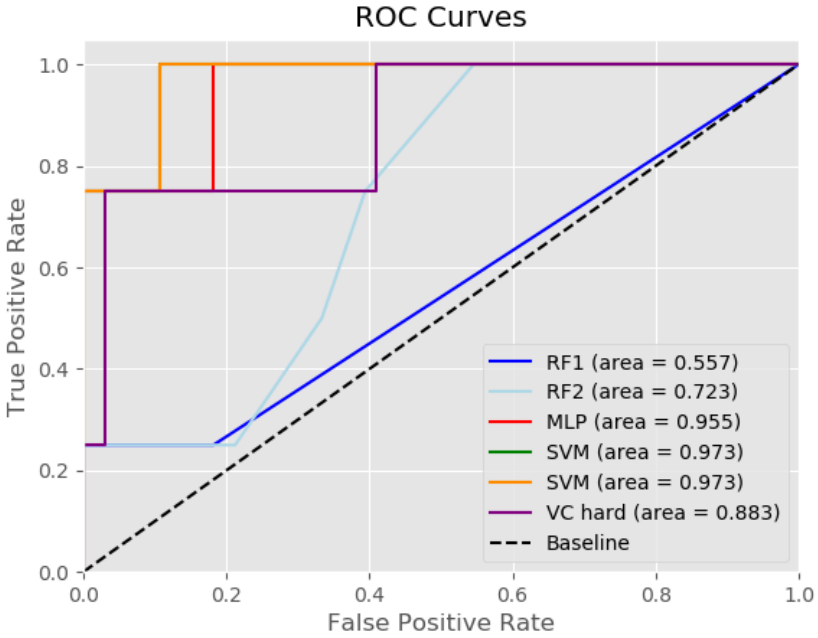

Fig. D.19. Phone 20: 70/30 Training/Test split

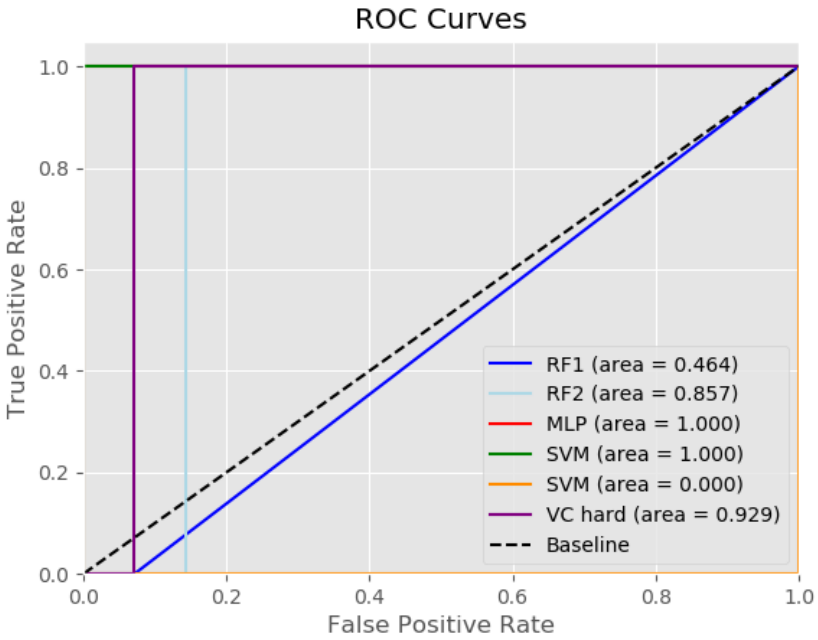

Fig. D.20. Phone 21: 70/30 Training/Test split

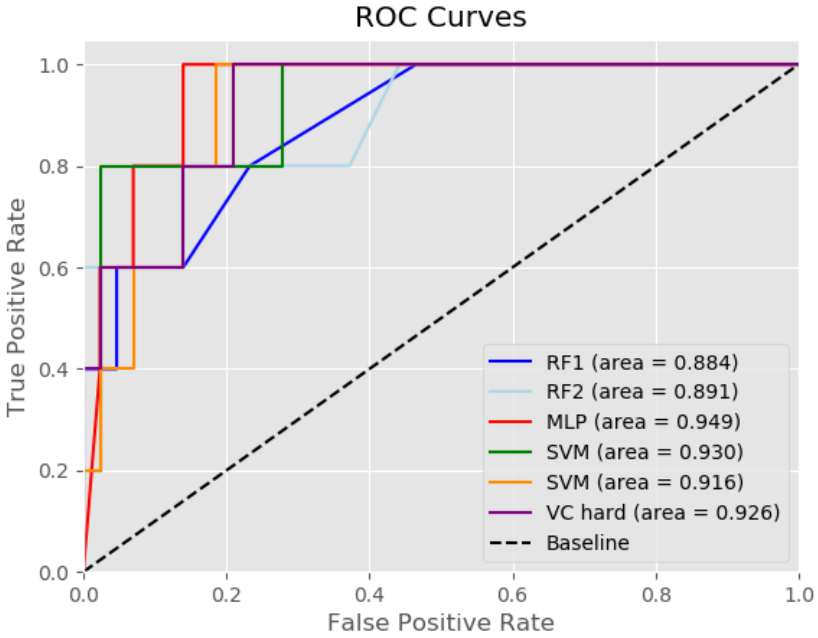

Fig. D.21. Phone 23: 70/30 Training/Test split

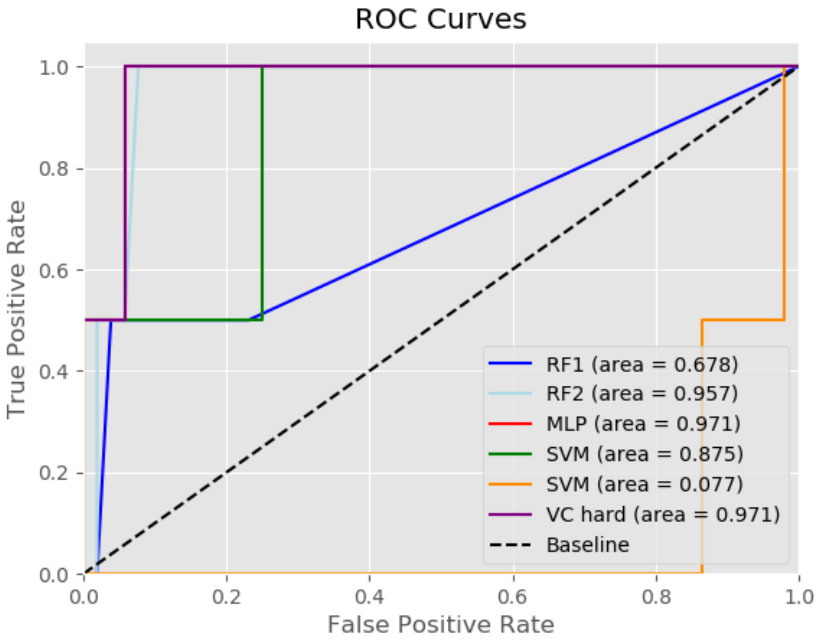

Fig. D.22. Phone 24: 70/30 Training/Test split

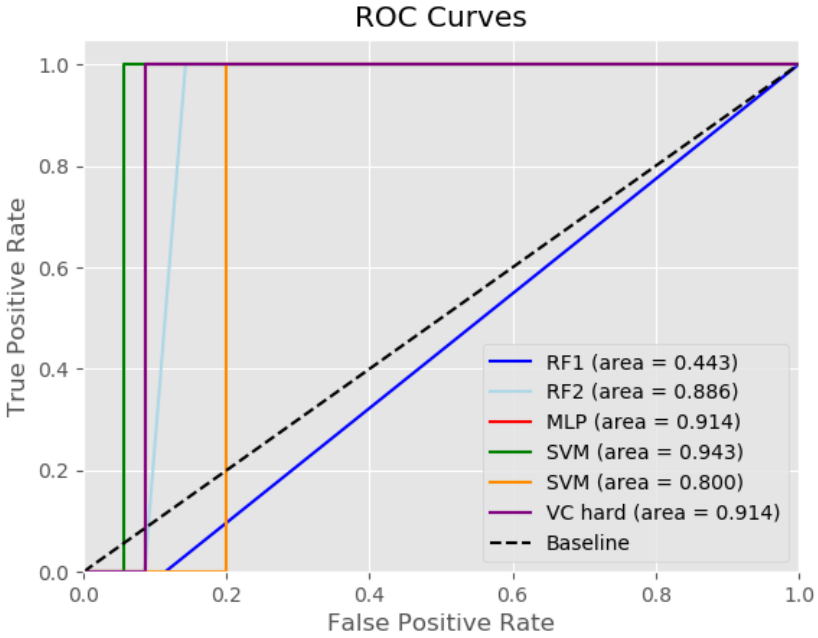

Fig. D.23. Phone 25: 70/30 Training/Test split

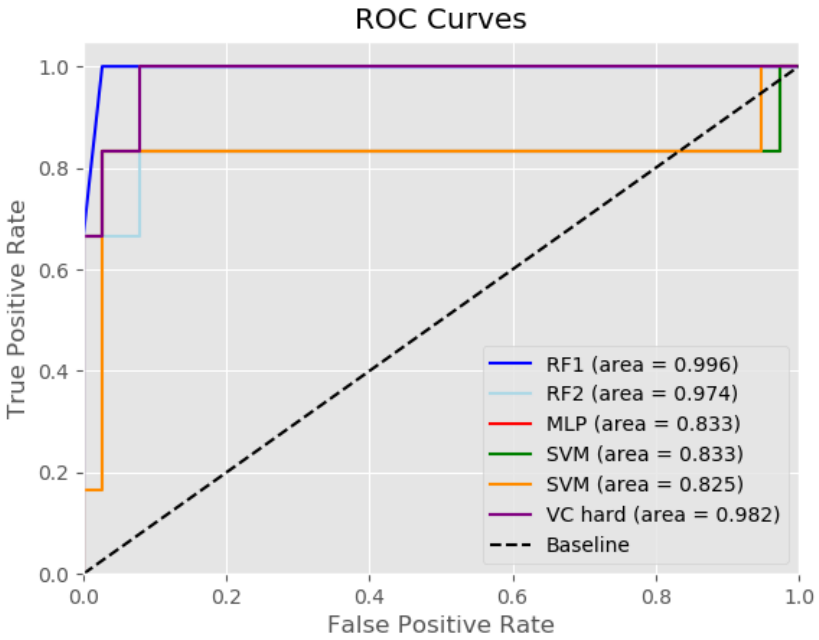

Fig. D.24. Phone 26: 70/30 Training/Test split

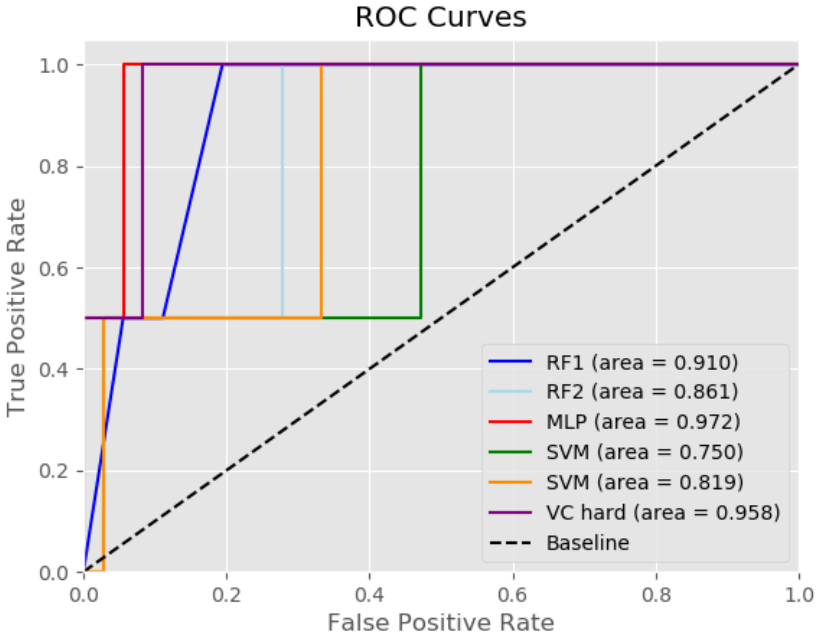

Fig. D.25. Phone 27: 70/30 Training/Test split

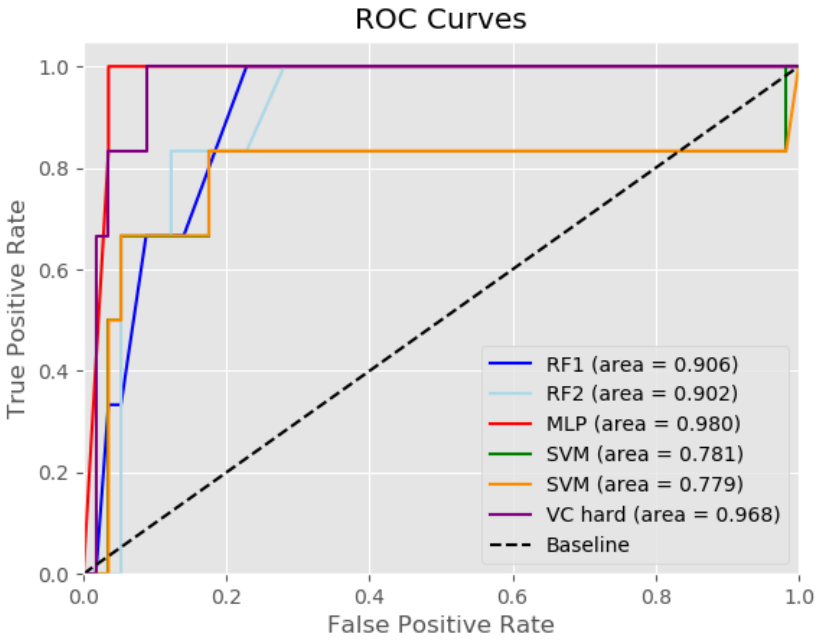

Fig. D.26. Phone 28: 70/30 Training/Test split

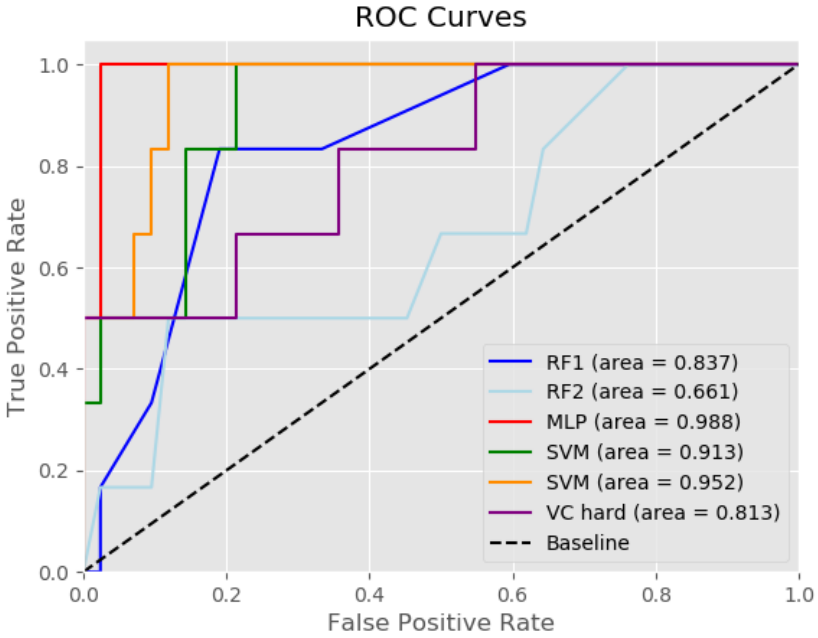

Fig. D.27. Phone 29: 70/30 Training/Test split

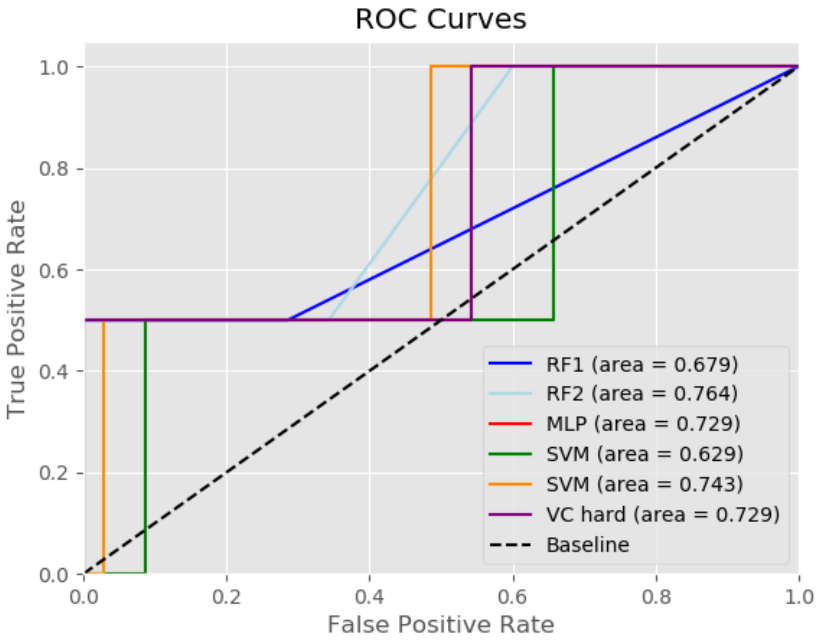

Fig. D.28. Phone 30: 70/30 Training/Test split

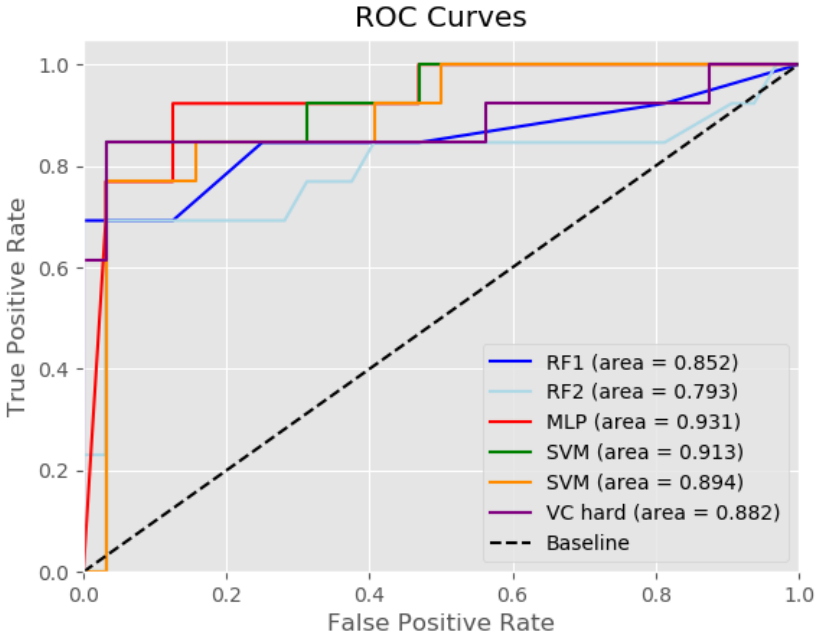

Fig. D.29. Phone 31: 70/30 Training/Test split

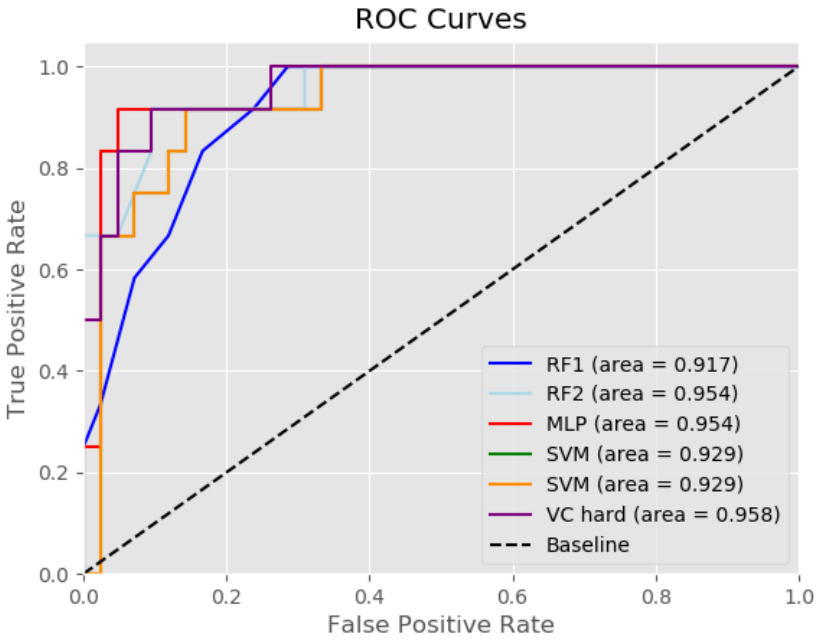

Fig. D.30. Phone 32: 70/30 Training/Test split

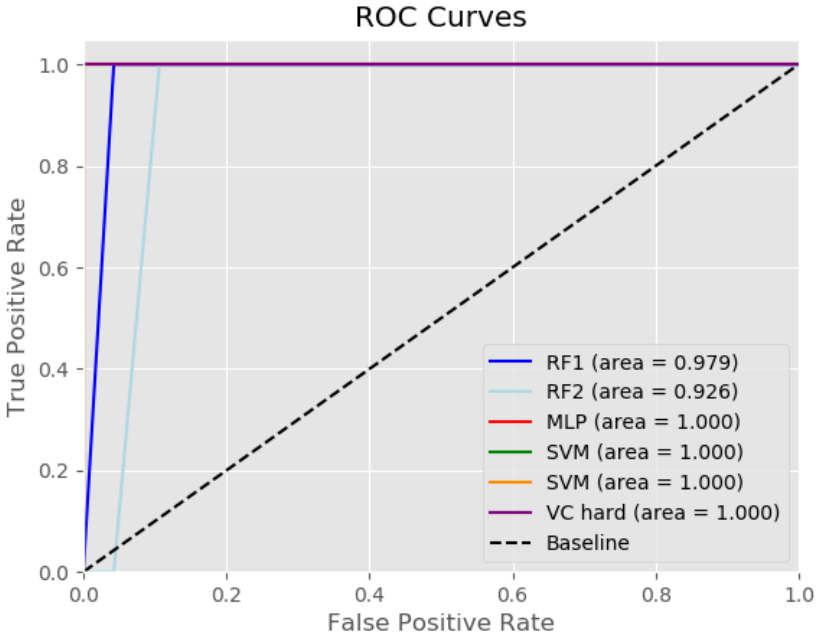

Fig. D.31. Phone 34: 70/30 Training/Test split

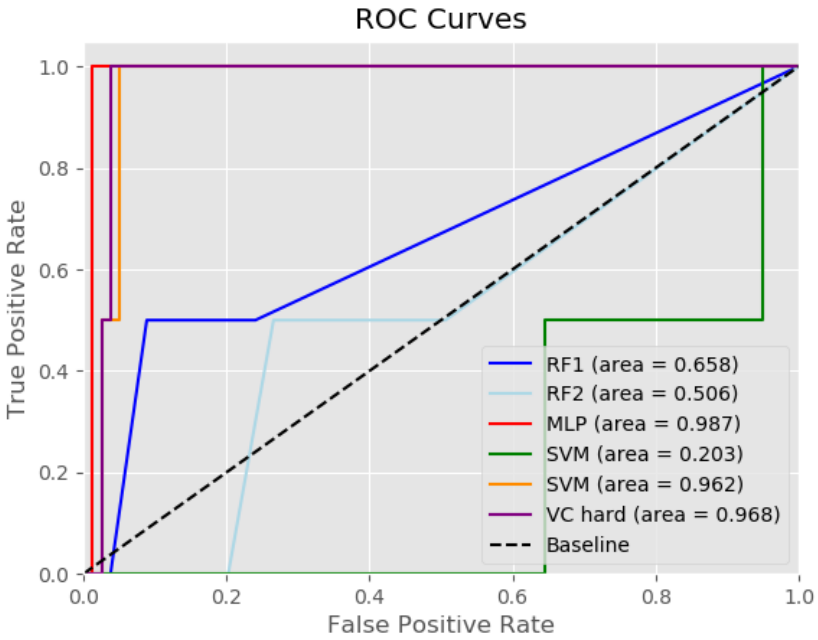

Fig. D.32. Phone 35: 70/30 Training/Test split

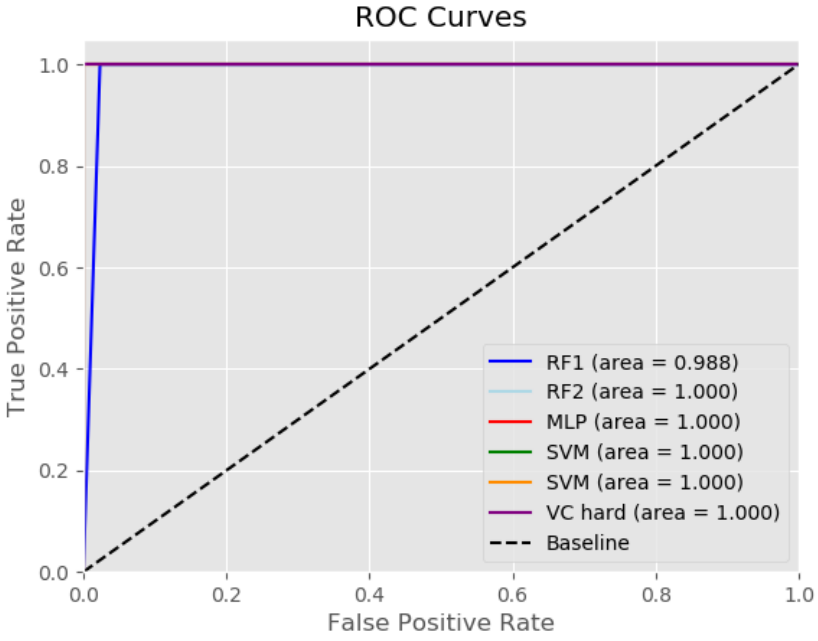

Fig. D.33. Phone 36: 70/30 Training/Test split

E. ROC CURVES 30/70 TRAINING TEST SPLIT

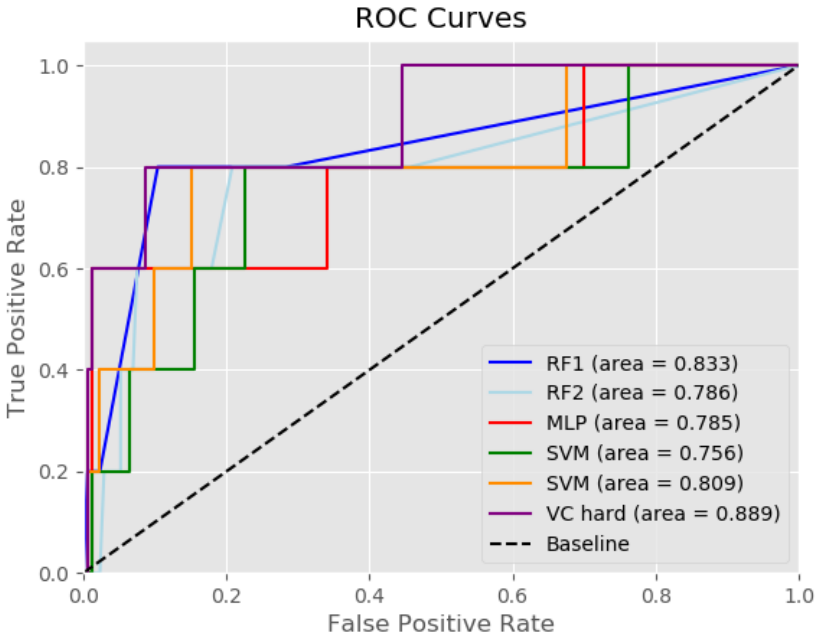

Fig. E.1. Phone 1: 30/70 Training/Test split

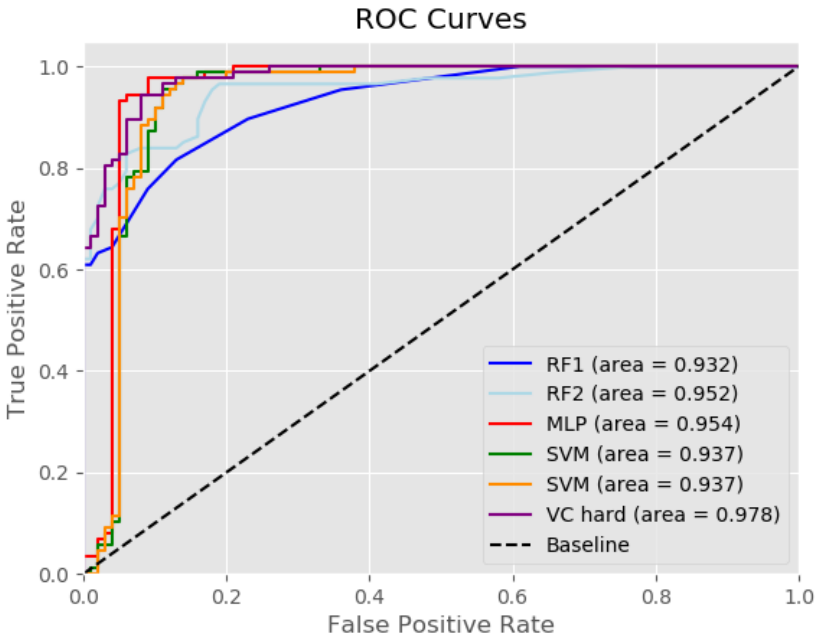

Fig. E.2. Phone 3: 30/70 Training/Test split

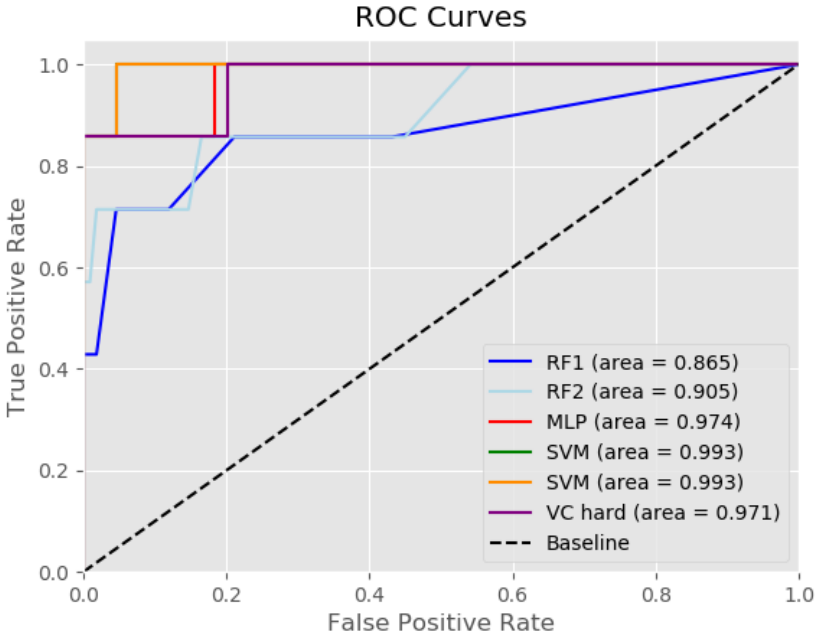

Fig. E.3. Phone 4: 30/70 Training/Test split

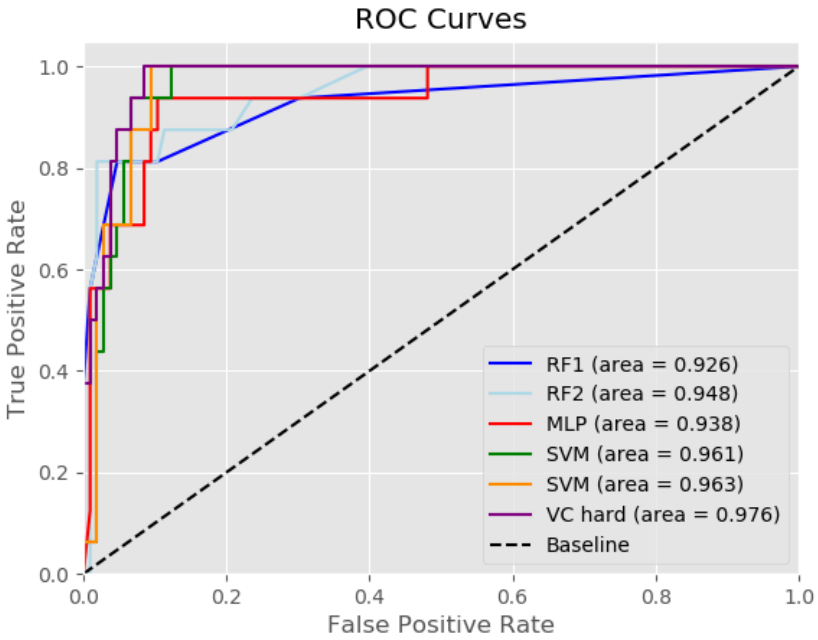

Fig. E.4. Phone 5: 30/70 Training/Test split

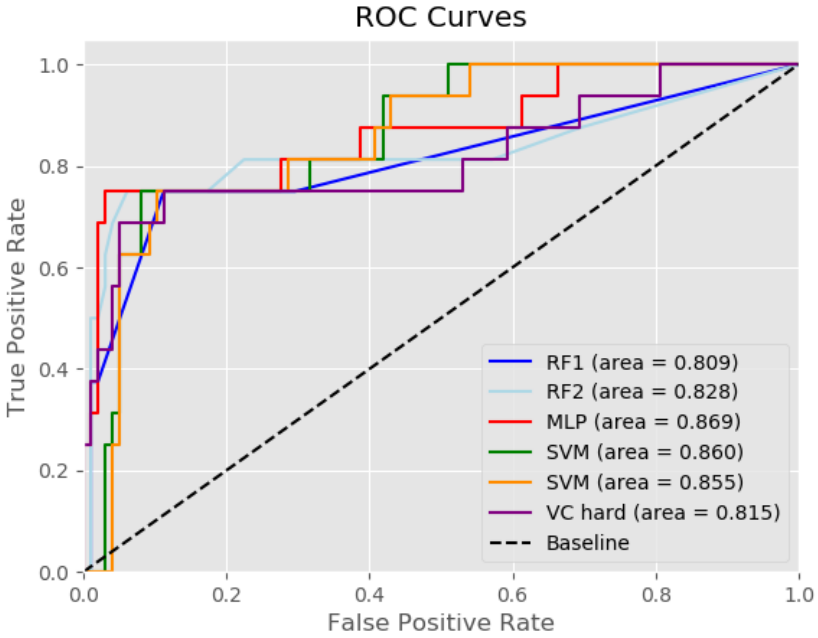

Fig. E.5. Phone 6: 30/70 Training/Test split

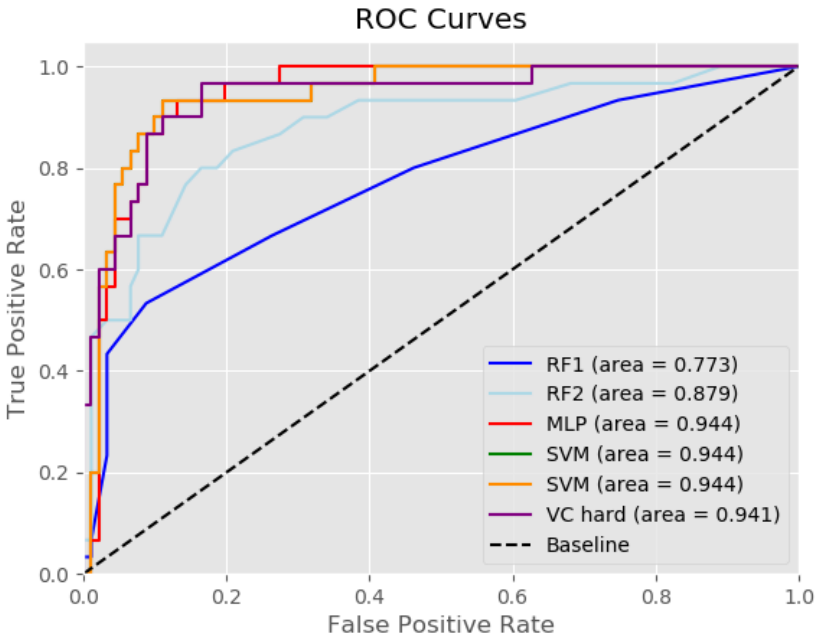

Fig. E.6. Phone 7: 30/70 Training/Test split

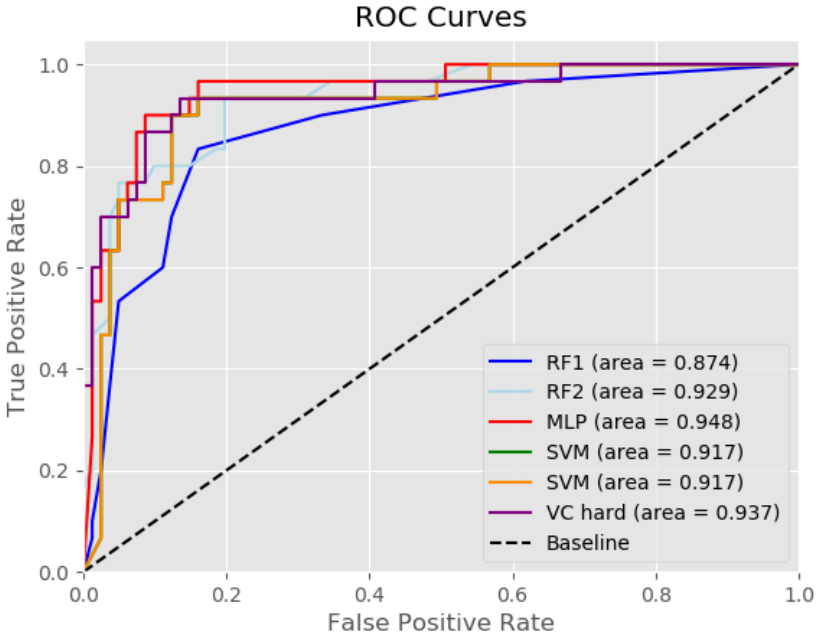

Fig. E.7. Phone 8: 30/70 Training/Test split

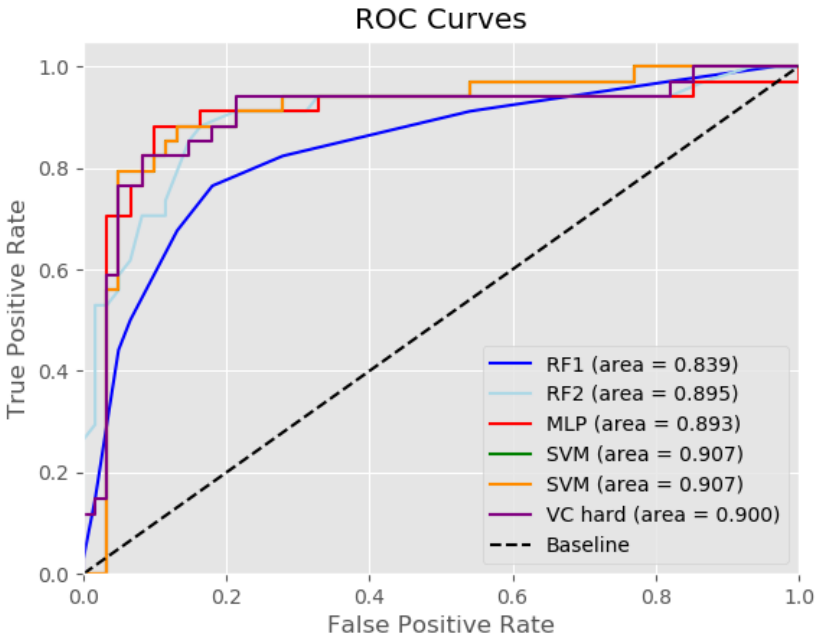

Fig. E.8. Phone 9: 30/70 Training/Test split

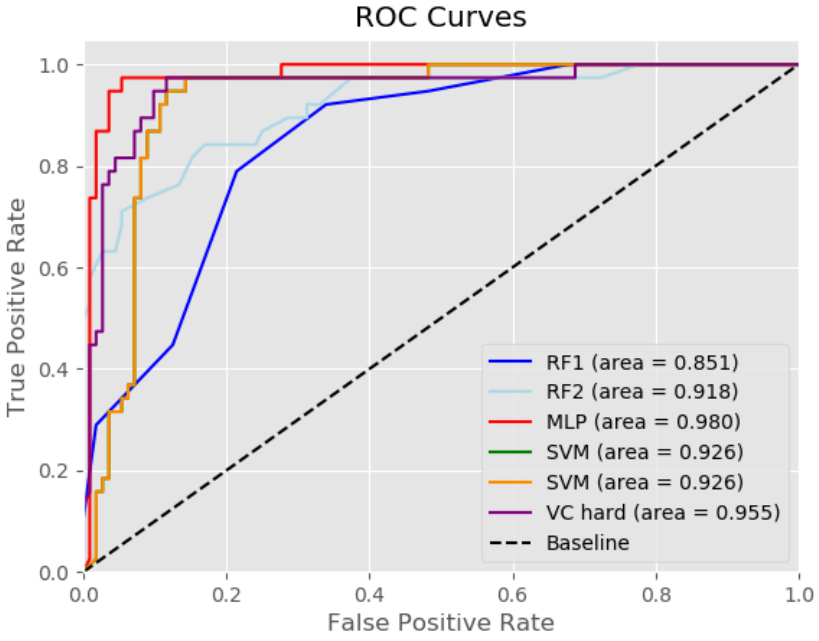

Fig. E.9. Phone 10: 30/70 Training/Test split

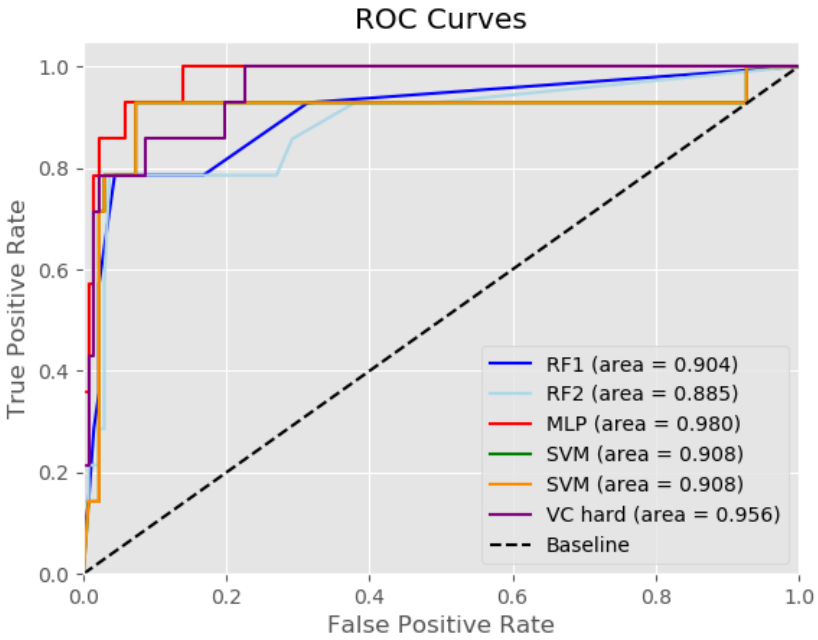

Fig. E.10. Phone 11: 30/70 Training/Test split

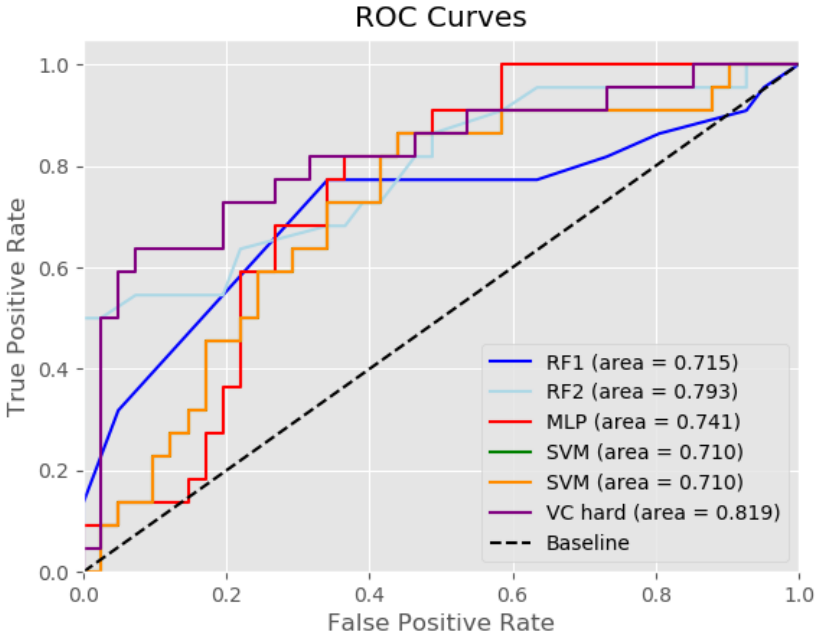

Fig. E.11. Phone 12: 30/70 Training/Test split

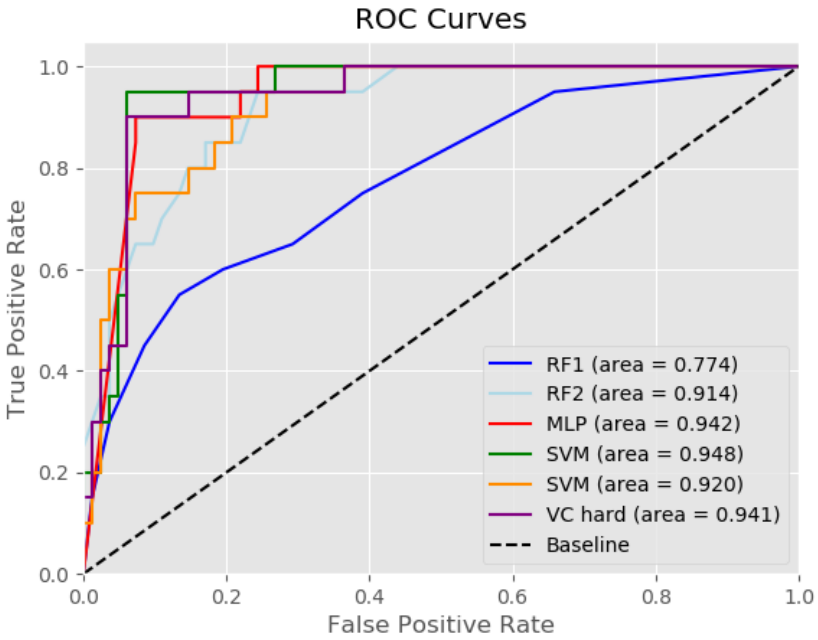

Fig. E.12. Phone 13: 30/70 Training/Test split

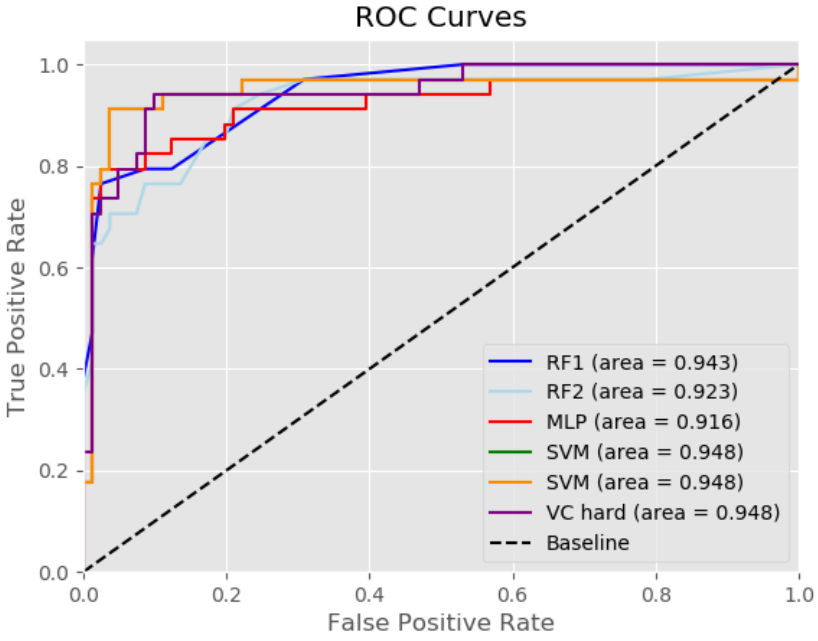

Fig. E.13. Phone 14: 30/70 Training/Test split

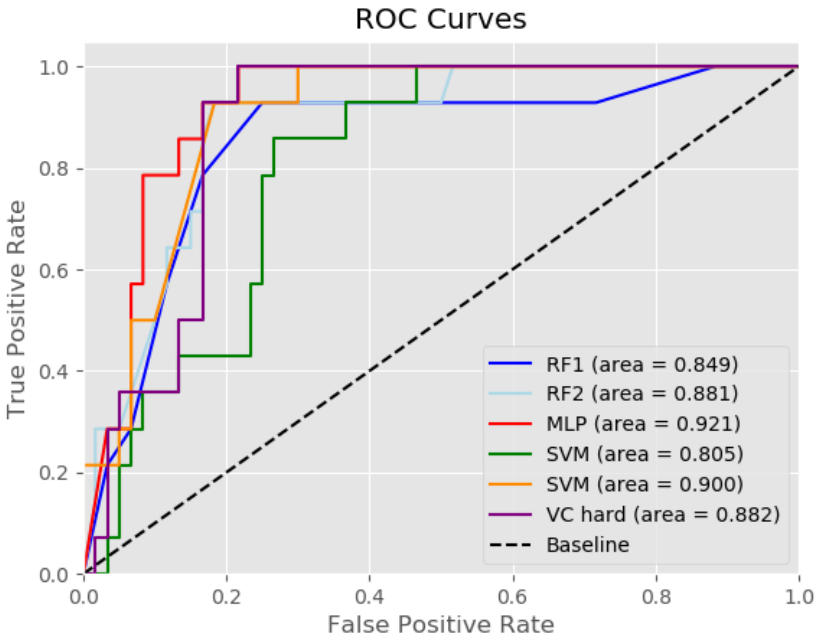

Fig. E.14. Phone 15: 30/70 Training/Test split

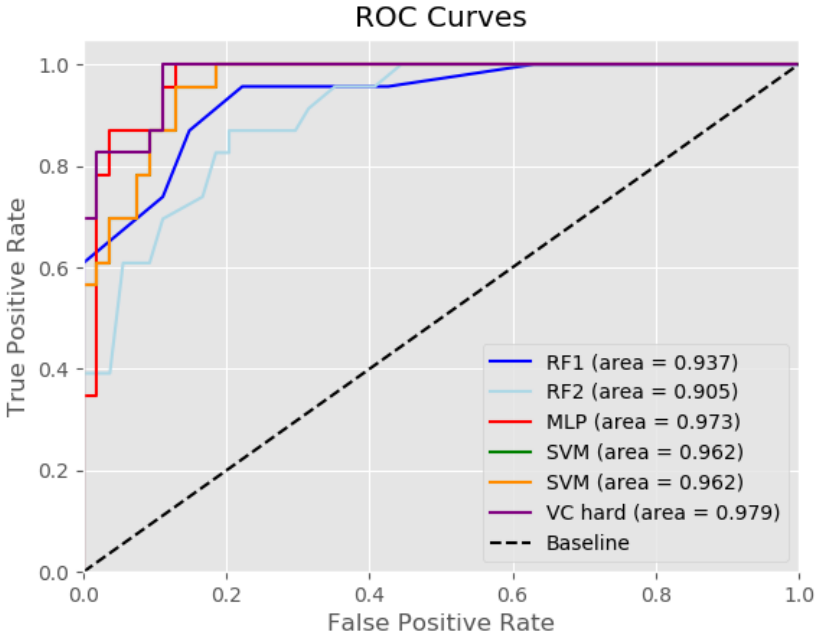

Fig. E.15. Phone 16: 30/70 Training/Test split

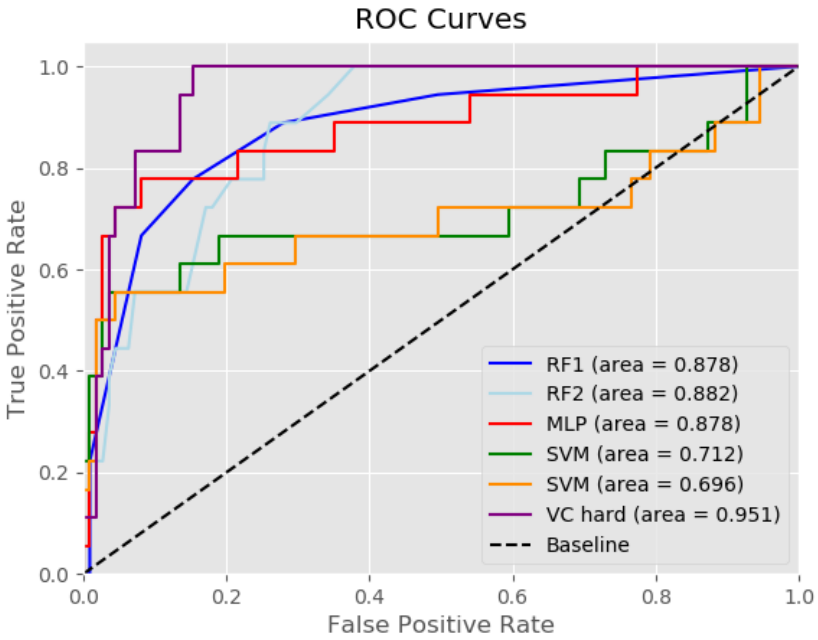

Fig. E.16. Phone 17: 30/70 Training/Test split

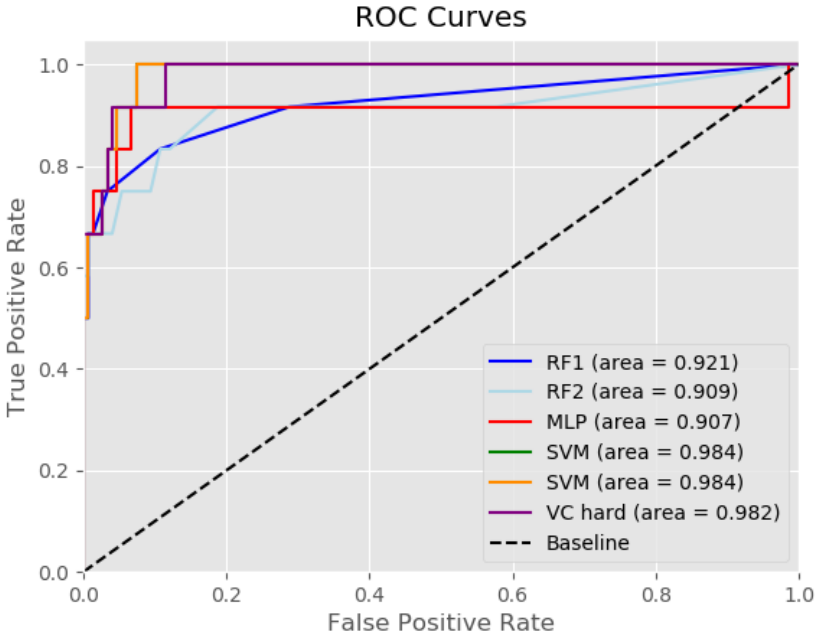

Fig. E.17. Phone 18: 30/70 Training/Test split

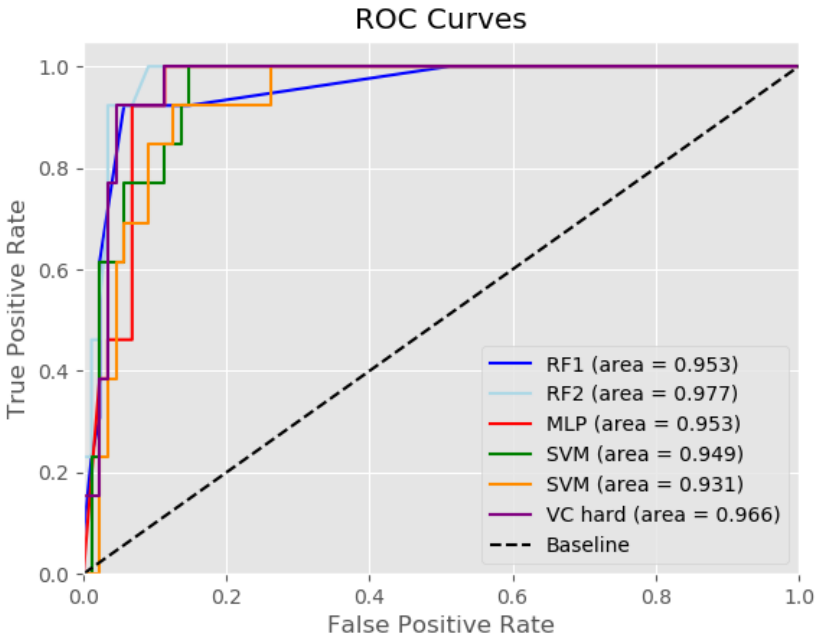

Fig. E.18. Phone 19: 30/70 Training/Test split

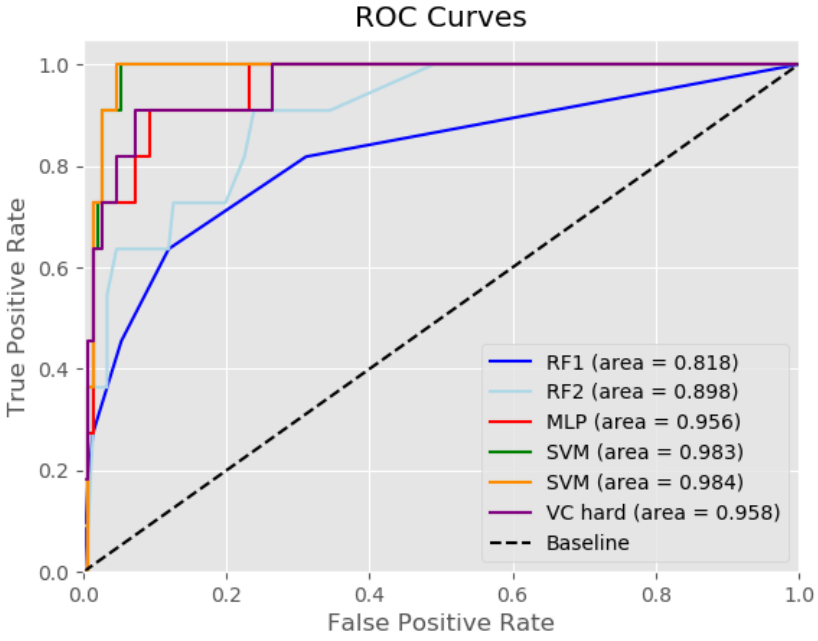

Fig. E.19. Phone 20: 30/70 Training/Test split

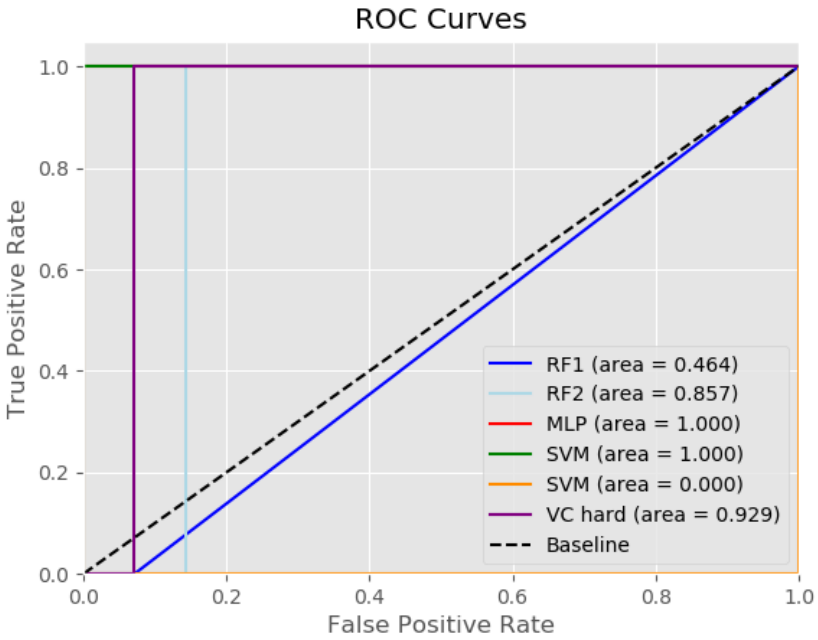

Fig. E.20. Phone 21: 30/70 Training/Test split

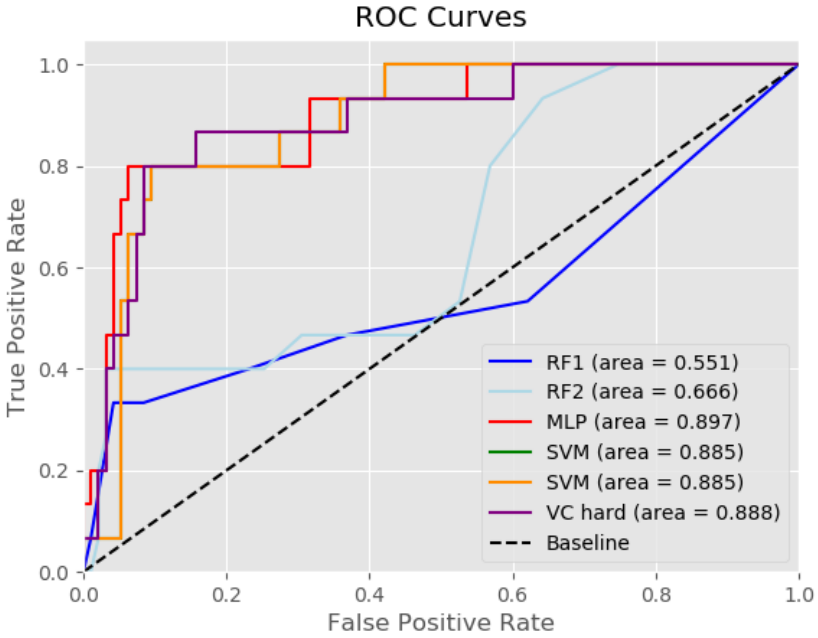

Fig. E.21. Phone 23: 30/70 Training/Test split

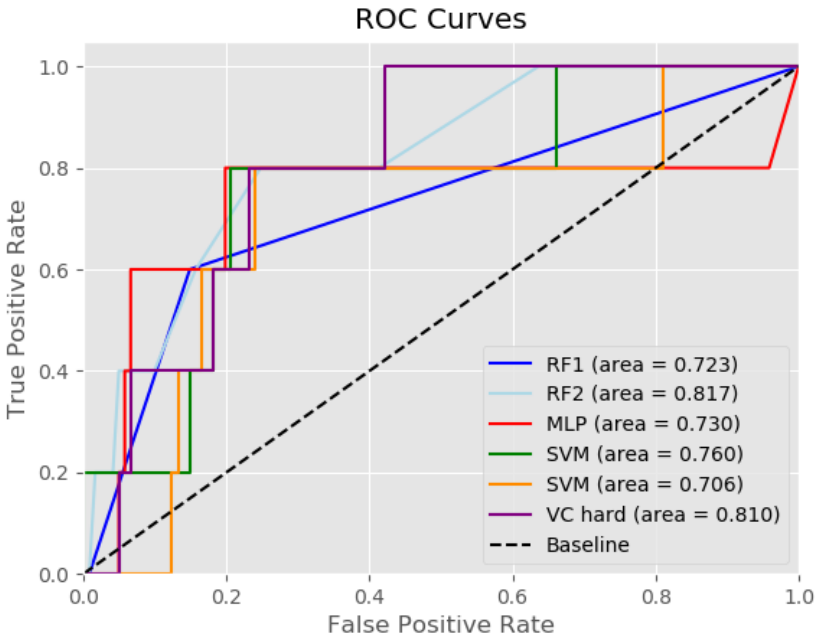

Fig. E.22. Phone 24: 30/70 Training/Test split

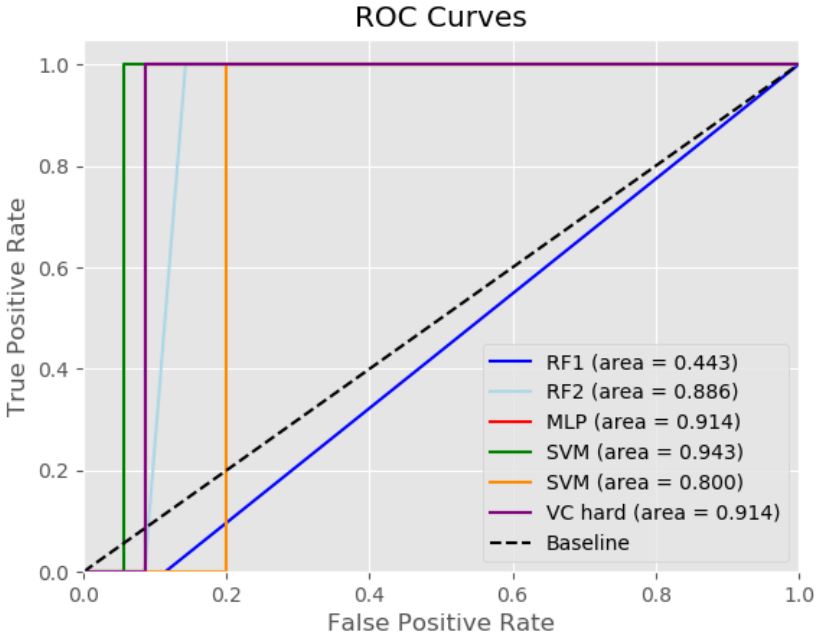

Fig. E.23. Phone 25: 30/70 Training/Test split

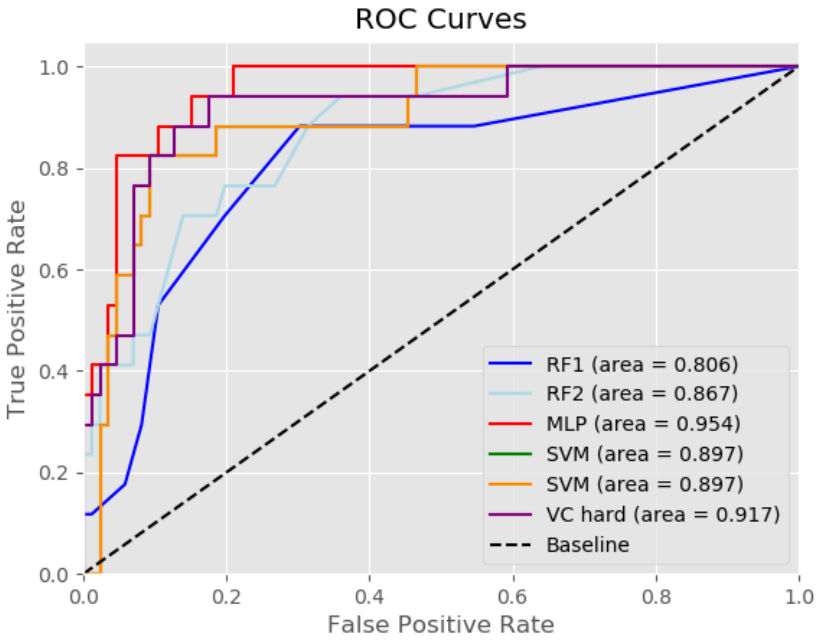

Fig. E.24. Phone 26: 30/70 Training/Test split

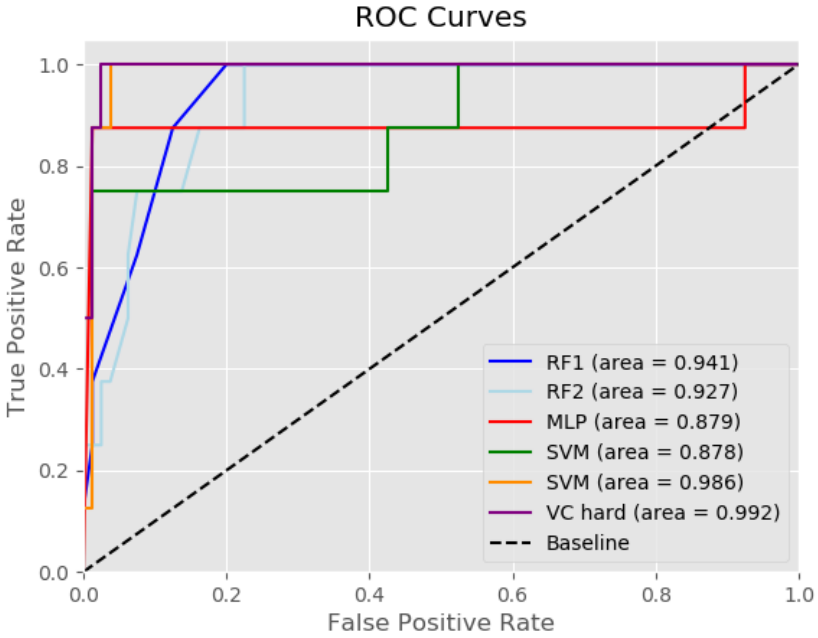

Fig. E.25. Phone 27: 30/70 Training/Test split

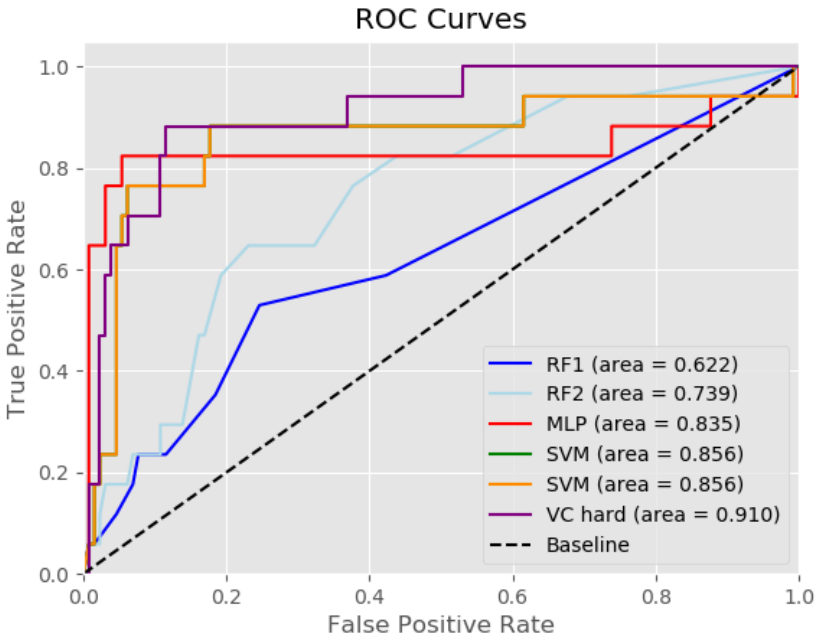

Fig. E.26. Phone 28: 30/70 Training/Test split

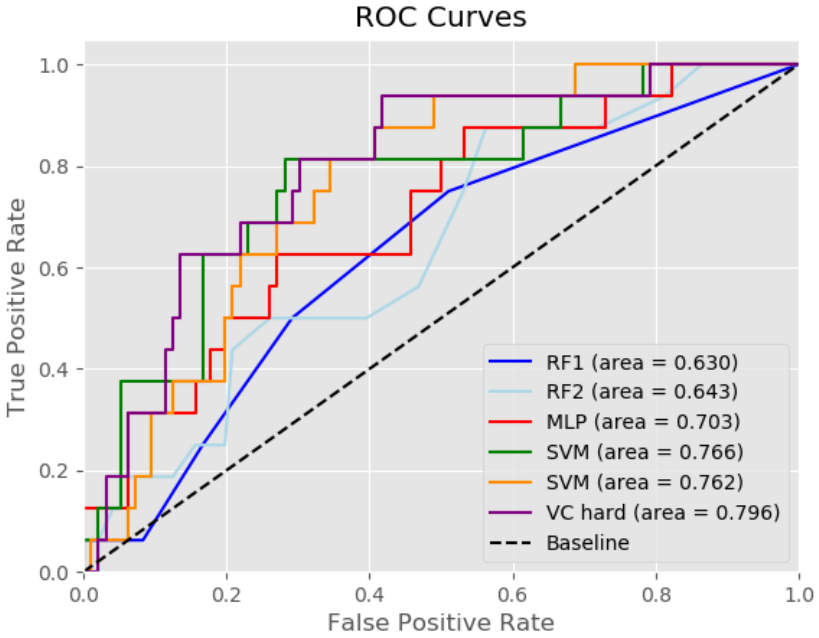

Fig. E.27. Phone 29: 30/70 Training/Test split

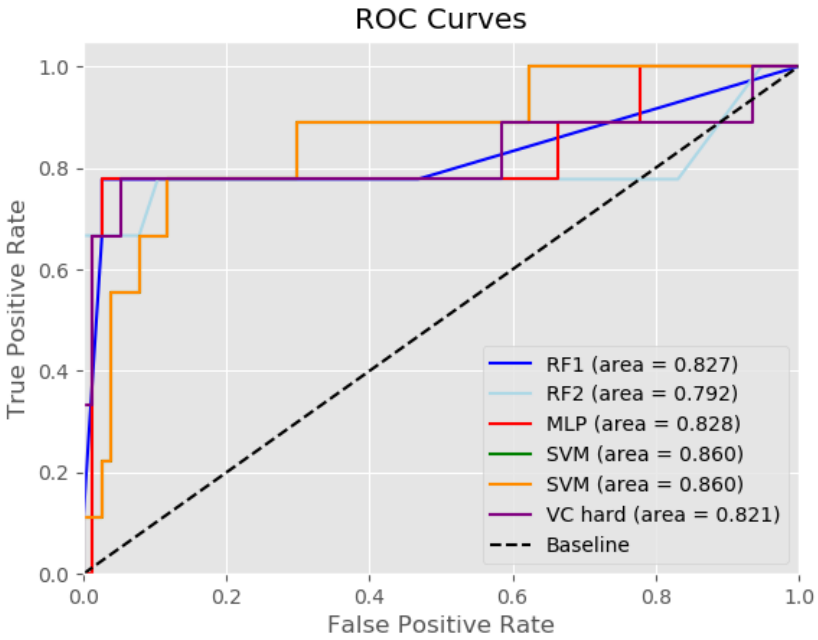

Fig. E.28. Phone 30: 30/70 Training/Test split

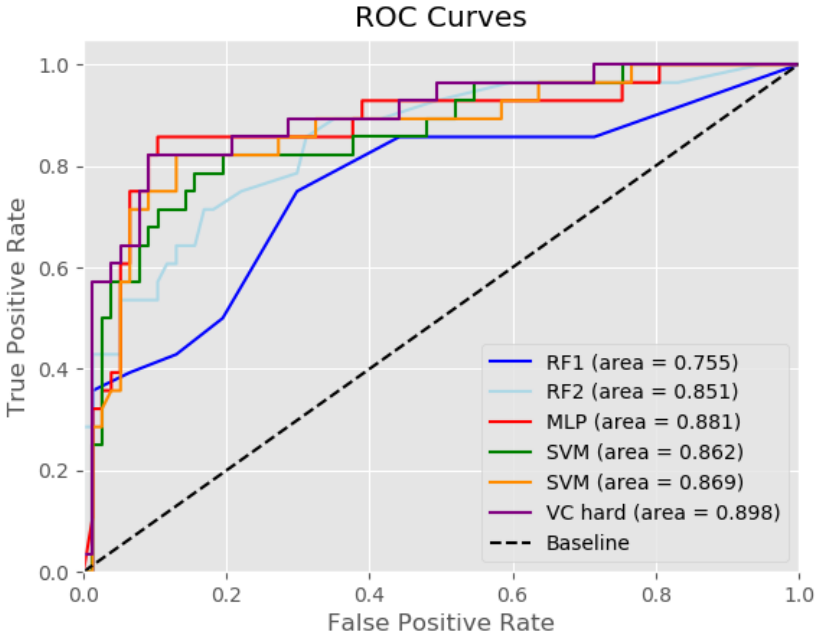

Fig. E.29. Phone 31: 30/70 Training/Test split

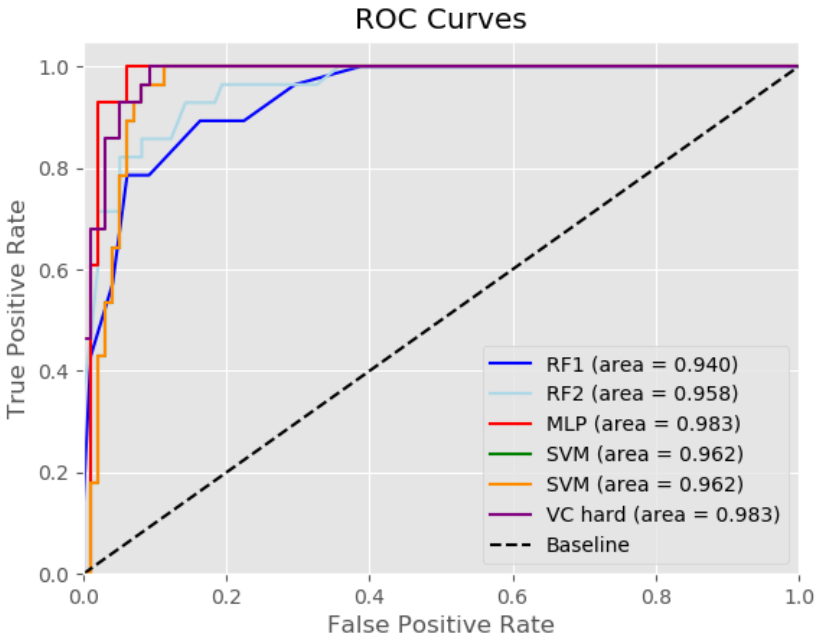

Fig. E.30. Phone 32: 30/70 Training/Test split

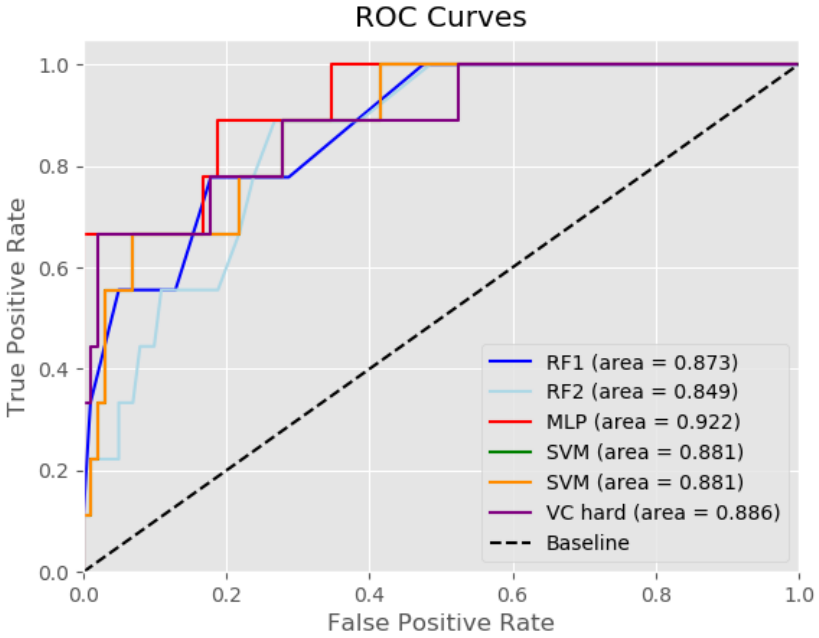

Fig. E.31. Phone 34: 30/70 Training/Test split

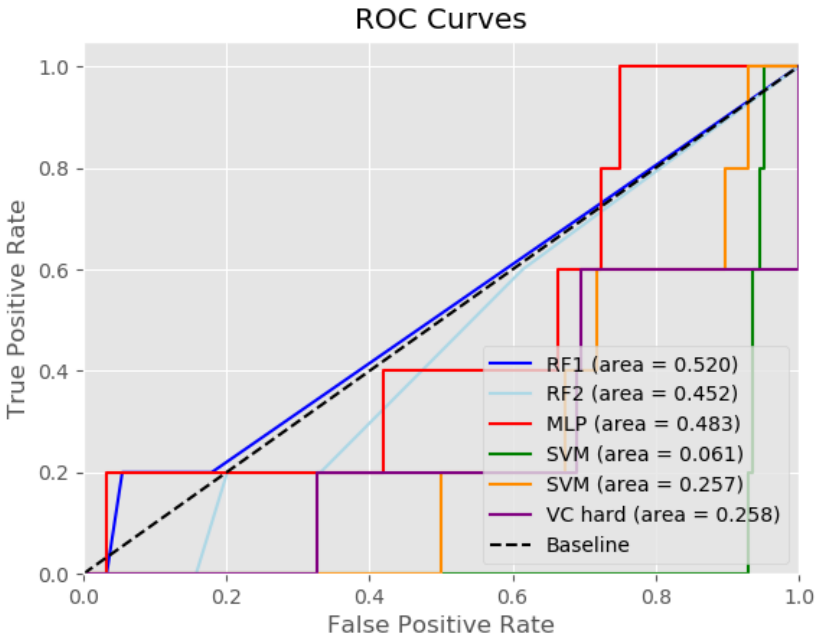

Fig. E.32. Phone 35: 30/70 Training/Test split

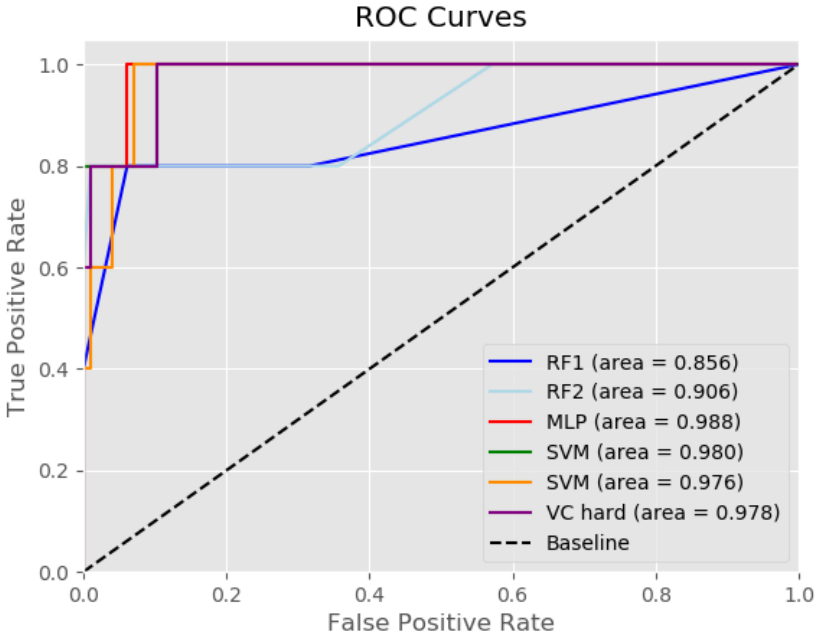

Fig. E.33. Phone 36: 30/70 Training/Test split

F. TWO-CLASS STATISTICS FOR INDIVIDUAL PHONES

Phone 1: 80/20

PCA Number of Variables: 39

Model Results:

SVM 1 Accuracy (C=1): 0.961 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.951 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.941 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.907 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.922

Random Forest Score: 0.9215686274509803

Random Forest (50) mean accuracy score: 0.922

Random Forest Score: 0.9215686274509803

MLP mean accuracy mean accuracy: 0.961

MLP Score: 0.9607843137254902

VC mean accuracy: 0.961

VC Score: 0.9607843137254902

Phone 1: 70/30

PCA Number of Variables: 36

Model Results:

SVM 1 Accuracy (C=1): 0.935 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.938 (+/- 0.06)

SVM 2 Accuracy (C=20): 0.935 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.956 (+/- 0.07)

Random Forest (10) mean accuracy score: 0.948

Random Forest Score: 0.948051948051948

Random Forest (50) mean accuracy score: 0.948

Random Forest Score: 0.948051948051948

MLP mean accuracy mean accuracy: 0.961

MLP Score: 0.961038961038961

VC mean accuracy: 0.974

VC Score: 0.974025974025974

Phone 1: 30/70

PCA Number of Variables: 23

Model Results:

SVM 1 Accuracy (C=1): 0.927 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.921 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.949 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.871 (+/- 0.25)

Random Forest (10) mean accuracy score: 0.972

Random Forest Score: 0.9719101123595506

Random Forest (50) mean accuracy score: 0.972

Random Forest Score: 0.9719101123595506

MLP mean accuracy mean accuracy: 0.966

MLP Score: 0.9662921348314607

VC mean accuracy: 0.972

VC Score: 0.9719101123595506

Phone 3: 80/20

PCA Number of Variables: 38

Model Results:

SVM 1 Accuracy (C=1): 0.963 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.953 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.963 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.930 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.926

Random Forest Score: 0.9259259259259259

Random Forest (50) mean accuracy score: 0.926

Random Forest Score: 0.9259259259259259

MLP mean accuracy mean accuracy: 0.944

MLP Score: 0.9444444444444444

VC mean accuracy: 0.963

VC Score: 0.9629629629629629

Phone 3: 70/30

PCA Number of Variables: 37

Model Results:

SVM 1 Accuracy (C=1): 0.938 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.898 (+/- 0.12)

SVM 2 Accuracy (C=20): 0.938 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.926 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.901

Random Forest Score: 0.9012345679012346

Random Forest (50) mean accuracy score: 0.938

Random Forest Score: 0.9382716049382716

MLP mean accuracy mean accuracy: 0.938

MLP Score: 0.9382716049382716

VC mean accuracy: 0.938

VC Score: 0.9382716049382716

Phone 3: 30/70

PCA Number of Variables: 24

Model Results:

SVM 1 Accuracy (C=1): 0.909 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.924 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.909 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.935 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.813

Random Forest Score: 0.8128342245989305

Random Forest (50) mean accuracy score: 0.850

Random Forest Score: 0.8502673796791443

MLP mean accuracy mean accuracy: 0.941

MLP Score: 0.9411764705882353

VC mean accuracy: 0.925

VC Score: 0.9251336898395722

Phone 4: 80/20

PCA Number of Variables: 29

Model Results:

SVM 1 Accuracy (C=1): 1.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.985 (+/- 0.03)

SVM 2 Accuracy (C=20): 1.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.977 (+/- 0.07)

Random Forest (10) mean accuracy score: 1.000

Random Forest Score: 1.0

Random Forest (50) mean accuracy score: 1.000

Random Forest Score: 1.0

MLP mean accuracy mean accuracy: 0.970

MLP Score: 0.9696969696969697

VC mean accuracy: 1.000

VC Score: 1.0

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.970 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.977 (+/- 0.07)

Random Forest (10) mean accuracy score: 0.000

Random Forest Score: 0.0

Random Forest (50) mean accuracy score: 0.600

Random Forest Score: 0.0

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 4: 70/30

PCA Number of Variables: 31

Model Results:

SVM 1 Accuracy (C=1): 1.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.974 (+/- 0.02)

SVM 2 Accuracy (C=20): 1.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.955 (+/- 0.12)

Random Forest (10) mean accuracy score: 1.000

Random Forest Score: 1.0

Random Forest (50) mean accuracy score: 1.000

Random Forest Score: 1.0

MLP mean accuracy mean accuracy: 0.980

MLP Score: 0.98

VC mean accuracy: 1.000

VC Score: 1.0

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.948 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.955 (+/- 0.12)

Random Forest (10) mean accuracy score: 0.200

Random Forest Score: 0.2

Random Forest (50) mean accuracy score: 0.200

Random Forest Score: 0.2

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 4: 30/70

PCA Number of Variables: 18

Model Results:

SVM 1 Accuracy (C=1): 0.974 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.980 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.974 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.967 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.966

Random Forest Score: 0.9655172413793104

Random Forest (50) mean accuracy score: 0.948

Random Forest Score: 0.9482758620689655

MLP mean accuracy mean accuracy: 0.974

MLP Score: 0.9741379310344828

VC mean accuracy: 0.983

Attack Session Results:

SVM 1 Accuracy (C=1): 0.200 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.959 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.200 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.967 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.200

Random Forest Score: 0.2

Random Forest (50) mean accuracy score: 0.400

Random Forest Score: 0.2

MLP mean accuracy mean accuracy: 0.200

MLP Score: 0.2

VC mean accuracy: 0.200

VC Score: 0.2

Phone 5: 80/20

PCA Number of Variables: 11

Model Results:

SVM 1 Accuracy (C=1): 0.877 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.865 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.934 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.867 (+/- 0.28)

Random Forest (10) mean accuracy score: 0.918

Random Forest Score: 0.9180327868852459

Random Forest (50) mean accuracy score: 0.885

Random Forest Score: 0.8852459016393442

MLP mean accuracy mean accuracy: 0.926

MLP Score: 0.9262295081967213

VC mean accuracy: 0.910

VC Score: 0.9098360655737705

Phone 5: 70/30

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 0.906 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.934 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.906 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.945 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.925

Random Forest Score: 0.9245283018867925

Random Forest (50) mean accuracy score: 0.925

Random Forest Score: 0.9245283018867925

MLP mean accuracy mean accuracy: 0.887

MLP Score: 0.8867924528301887

VC mean accuracy: 0.906

VC Score: 0.9056603773584906

Phone 5: 30/70

PCA Number of Variables: 11

Model Results:

SVM 1 Accuracy (C=1): 0.877 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.865 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.934 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.867 (+/- 0.28)

Random Forest (10) mean accuracy score: 0.918

Random Forest Score: 0.9180327868852459

Random Forest (50) mean accuracy score: 0.885

Random Forest Score: 0.8852459016393442

MLP mean accuracy mean accuracy: 0.926

MLP Score: 0.9262295081967213

VC mean accuracy: 0.910

VC Score: 0.9098360655737705

Phone 6: 80/20

PCA Number of Variables: 28

Model Results:

SVM 1 Accuracy (C=1): 0.939 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.892 (+/- 0.09)

SVM 2 Accuracy (C=20): 0.879 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.867 (+/- 0.19)

Random Forest (10) mean accuracy score: 0.909

Random Forest Score: 0.9090909090909091

Random Forest (50) mean accuracy score: 0.939

Random Forest Score: 0.9393939393939394

MLP mean accuracy mean accuracy: 0.879

MLP Score: 0.8787878787878788

VC mean accuracy: 0.939

VC Score: 0.9393939393939394

Phone 6: 70/30

PCA Number of Variables: 27

Model Results:

SVM 1 Accuracy (C=1): 0.939 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.876 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.857 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.864 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.959

Random Forest Score: 0.9591836734693877

Random Forest (50) mean accuracy score: 0.959

Random Forest Score: 0.9591836734693877

MLP mean accuracy mean accuracy: 0.857

MLP Score: 0.8571428571428571

VC mean accuracy: 0.959

VC Score: 0.9591836734693877

Phone 6: 30/70

PCA Number of Variables: 15 Model Results:

SVM 1 Accuracy (C=1): 0.895 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.960 (+/- 0.08)

SVM 2 Accuracy (C=20): 0.877 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.963 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.904

Random Forest Score: 0.9035087719298246

Random Forest (50) mean accuracy score: 0.904

Random Forest Score: 0.9035087719298246

MLP mean accuracy mean accuracy: 0.930

MLP Score: 0.9298245614035088

VC mean accuracy: 0.895

VC Score: 0.8947368421052632

Phone 7: 80/20

PCA Number of Variables: 42

Model Results:

SVM 1 Accuracy (C=1): 0.886 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.920 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.857 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.928 (+/- 0.17)

Random Forest (10) mean accuracy score: 0.886

Random Forest Score: 0.8857142857142857

Random Forest (50) mean accuracy score: 0.914

Random Forest Score: 0.9142857142857143

MLP mean accuracy mean accuracy: 0.943

MLP Score: 0.9428571428571428

VC mean accuracy: 0.914

VC Score: 0.9142857142857143

Phone 7: 70/30

PCA Number of Variables: 39

Model Results:

SVM 1 Accuracy (C=1): 0.904 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.908 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.904 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.925 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.865

Random Forest Score: 0.8653846153846154

Random Forest (50) mean accuracy score: 0.885

Random Forest Score: 0.8846153846153846

MLP mean accuracy mean accuracy: 0.923

MLP Score: 0.9230769230769231

VC mean accuracy: 0.904

VC Score: 0.9038461538461539

Phone 7: 30/70

PCA Number of Variables: 23

Model Results:

SVM 1 Accuracy (C=1): 0.909 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.942 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.909 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.908 (+/- 0.19)

Random Forest (10) mean accuracy score: 0.785

Random Forest Score: 0.7851239669421488

Random Forest (50) mean accuracy score: 0.826

Random Forest Score: 0.8264462809917356

MLP mean accuracy mean accuracy: 0.884

MLP Score: 0.8842975206611571

VC mean accuracy: 0.876

VC Score: 0.8760330578512396

Phone 8: 80/20

PCA Number of Variables: 29

Model Results:

SVM 1 Accuracy (C=1): 0.812 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.881 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.844 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.901 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.844

Random Forest Score: 0.84375

Random Forest (50) mean accuracy score: 0.812

Random Forest Score: 0.8125

MLP mean accuracy mean accuracy: 0.906

MLP Score: 0.90625

VC mean accuracy: 0.844

VC Score: 0.84375

Phone 8: 70/30

PCA Number of Variables: 28

Model Results:

SVM 1 Accuracy (C=1): 0.854 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.872 (+/- 0.08)

SVM 2 Accuracy (C=20): 0.833 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.909 (+/- 0.14)

Random Forest (10) mean accuracy score: 0.812

Random Forest Score: 0.8125

Random Forest (50) mean accuracy score: 0.896

Random Forest Score: 0.8958333333333334

MLP mean accuracy mean accuracy: 0.896

MLP Score: 0.8958333333333334

VC mean accuracy: 0.833

VC Score: 0.8333333333333334

Phone 8: 30/70

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 0.865 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.808 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.865 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.860 (+/- 0.44)

Random Forest (10) mean accuracy score: 0.838

Random Forest Score: 0.8378378378378378

Random Forest (50) mean accuracy score: 0.874

Random Forest Score: 0.8738738738738738

MLP mean accuracy mean accuracy: 0.910

MLP Score: 0.9099099099099099

VC mean accuracy: 0.874

VC Score: 0.8738738738738738

Phone 9: 80/20

PCA Number of Variables: 30

Model Results:

SVM 1 Accuracy (C=1): 0.963 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.917 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.889 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.877 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.741

Random Forest Score: 0.7407407407407407

Random Forest (50) mean accuracy score: 0.778

Random Forest Score: 0.7777777777777778

MLP mean accuracy mean accuracy: 0.926

MLP Score: 0.9259259259259259

VC mean accuracy: 0.926

VC Score: 0.9259259259259259

Phone 9: 70/30

PCA Number of Variables: 29

Model Results:

SVM 1 Accuracy (C=1): 0.878 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.936 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.829 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.853 (+/- 0.21)

Random Forest (10) mean accuracy score: 0.829

Random Forest Score: 0.8292682926829268

Random Forest (50) mean accuracy score: 0.780

Random Forest Score: 0.7804878048780488

MLP mean accuracy mean accuracy: 0.854

MLP Score: 0.8536585365853658

VC mean accuracy: 0.854

VC Score: 0.8536585365853658

Phone 9: 30/70

PCA Number of Variables: 20

Model Results:

SVM 1 Accuracy (C=1): 0.863 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.776 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.863 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.893 (+/- 0.27)

Random Forest (10) mean accuracy score: 0.800

Random Forest Score: 0.8

Random Forest (50) mean accuracy score: 0.821

Random Forest Score: 0.8210526315789474

MLP mean accuracy mean accuracy: 0.874

MLP Score: 0.8736842105263158

VC mean accuracy: 0.863

VC Score: 0.8631578947368421

Phone 10: 80/20

PCA Number of Variables: 35

Model Results:

SVM 1 Accuracy (C=1): 0.907 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.959 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.907 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.930 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.907

Random Forest Score: 0.9069767441860465

Random Forest (50) mean accuracy score: 0.907

Random Forest Score: 0.9069767441860465

MLP mean accuracy mean accuracy: 0.907

MLP Score: 0.9069767441860465

VC mean accuracy: 0.930

VC Score: 0.9302325581395349

Phone 10: 70/30

PCA Number of Variables: 36

Model Results:

SVM 1 Accuracy (C=1): 0.908 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.966 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.908 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.960 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.877

Random Forest Score: 0.8769230769230769

Random Forest (50) mean accuracy score: 0.892

Random Forest Score: 0.8923076923076924

MLP mean accuracy mean accuracy: 0.908

MLP Score: 0.9076923076923077

VC mean accuracy: 0.892

VC Score: 0.8923076923076924

Phone 10: 30/70

PCA Number of Variables: 27

Model Results:

SVM 1 Accuracy (C=1): 0.900 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.861 (+/- 0.09)

SVM 2 Accuracy (C=20): 0.900 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.957 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.767

Random Forest Score: 0.7666666666666667

Random Forest (50) mean accuracy score: 0.867

Random Forest Score: 0.8666666666666667

MLP mean accuracy mean accuracy: 0.953

MLP Score: 0.9533333333333334

VC mean accuracy: 0.913

VC Score: 0.9133333333333333

Phone 11: 80/20

PCA Number of Variables: 37

Model Results:

SVM 1 Accuracy (C=1): 0.977

SVM 1 Accuracy cross validated (cv=2): 0.924 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.930 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.921 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.930

Random Forest Score: 0.9302325581395349

Random Forest (50) mean accuracy score: 0.930

Random Forest Score: 0.9302325581395349

MLP mean accuracy mean accuracy: 0.953

MLP Score: 0.9534883720930233

VC mean accuracy: 0.930

VC Score: 0.9302325581395349

Phone 11: 70/30

PCA Number of Variables: 35

Model Results:

SVM 1 Accuracy (C=1): 0.985 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.927 (+/- 0.09)

SVM 2 Accuracy (C=20): 0.954 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.922 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.969

Random Forest Score: 0.9692307692307692

Random Forest (50) mean accuracy score: 0.954

Random Forest Score: 0.9538461538461539

MLP mean accuracy mean accuracy: 0.969

MLP Score: 0.9692307692307692

VC mean accuracy: 0.954

VC Score: 0.9538461538461539

Phone 11: 30/70

PCA Number of Variables: 26

Model Results:

SVM 1 Accuracy (C=1): 0.934 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 1.000 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.934 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 1.000 (+/- 0.00)

Random Forest (10) mean accuracy score: 0.921

Random Forest Score: 0.9205298013245033

Random Forest (50) mean accuracy score: 0.907

Random Forest Score: 0.9072847682119205

MLP mean accuracy mean accuracy: 0.960

MLP Score: 0.9602649006622517

VC mean accuracy: 0.960

VC Score: 0.9602649006622517

Phone 12: 80/20

PCA Number of Variables: 24

Model Results:

SVM 1 Accuracy (C=1): 0.889 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.721 (+/- 0.07)

SVM 2 Accuracy (C=20): 0.944 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.720 (+/- 0.22)

Random Forest (10) mean accuracy score: 0.778

Random Forest Score: 0.7777777777777778

Random Forest (50) mean accuracy score: 0.778

Random Forest Score: 0.7777777777777778

MLP mean accuracy mean accuracy: 0.889

MLP Score: 0.8888888888888888

VC mean accuracy: 0.889

VC Score: 0.8888888888888888

Phone 12: 70/30

PCA Number of Variables: 24

Model Results:

SVM 1 Accuracy (C=1): 0.667 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.778 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.593 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.860 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.778

Random Forest Score: 0.7777777777777778

Random Forest (50) mean accuracy score: 0.630

Random Forest Score: 0.6296296296296297

MLP mean accuracy mean accuracy: 0.630

MLP Score: 0.6296296296296297

VC mean accuracy: 0.630

VC Score: 0.6296296296296297

Phone 12: 30/70

PCA Number of Variables: 14

Model Results:

SVM 1 Accuracy (C=1): 0.571 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.926 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.571 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.758 (+/- 0.52)

Random Forest (10) mean accuracy score: 0.587

Random Forest Score: 0.5873015873015873

Random Forest (50) mean accuracy score: 0.635

Random Forest Score: 0.6349206349206349

MLP mean accuracy mean accuracy: 0.667

MLP Score: 0.6666666666666666

VC mean accuracy: 0.651

VC Score: 0.6507936507936508

Phone 13: 80/20

PCA Number of Variables: 23

Model Results:

SVM 1 Accuracy (C=1): 0.862 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.931 (+/- 0.07)

SVM 2 Accuracy (C=20): 0.897 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.906 (+/- 0.14)

Random Forest (10) mean accuracy score: 0.897

Random Forest Score: 0.896551724137931

Random Forest (50) mean accuracy score: 0.862

Random Forest Score: 0.8620689655172413

MLP mean accuracy mean accuracy: 0.862

MLP Score: 0.8620689655172413

VC mean accuracy: 0.897

VC Score: 0.896551724137931

Phone 13: 70/30

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 0.909 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.910 (+/- 0.14)

SVM 2 Accuracy (C=20): 0.886 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.901 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.909

Random Forest Score: 0.9090909090909091

Random Forest (50) mean accuracy score: 0.909

Random Forest Score: 0.9090909090909091

MLP mean accuracy mean accuracy: 0.955

MLP Score: 0.9545454545454546

VC mean accuracy: 0.955

VC Score: 0.9545454545454546

Phone 13: 30/70

PCA Number of Variables: 12 Model Results:

SVM 1 Accuracy (C=1): 0.941 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.719 (+/- 0.20)

SVM 2 Accuracy (C=20): 0.882 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.797 (+/- 0.38)

Random Forest (10) mean accuracy score: 0.824

Random Forest Score: 0.8235294117647058

Random Forest (50) mean accuracy score: 0.873

Random Forest Score: 0.8725490196078431

MLP mean accuracy mean accuracy: 0.843

MLP Score: 0.8431372549019608

VC mean accuracy: 0.931

VC Score: 0.9313725490196079

Phone 14: 80/20

PCA Number of Variables: 32

Model Results:

SVM 1 Accuracy (C=1): 0.879 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.916 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.970 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.879 (+/- 0.19)

Random Forest (10) mean accuracy score: 0.818

Random Forest Score: 0.8181818181818182

Random Forest (50) mean accuracy score: 0.879

Random Forest Score: 0.8787878787878788

MLP mean accuracy mean accuracy: 1.000

MLP Score: 1.0

VC mean accuracy: 0.970

VC Score: 0.9696969696969697

Phone 14: 70/30

PCA Number of Variables: 29

Model Results:

SVM 1 Accuracy (C=1): 0.920 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.930 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.920 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.848 (+/- 0.23)

Random Forest (10) mean accuracy score: 0.840

Random Forest Score: 0.84

Random Forest (50) mean accuracy score: 0.900

Random Forest Score: 0.9

MLP mean accuracy mean accuracy: 0.960

MLP Score: 0.96

VC mean accuracy: 0.940

VC Score: 0.94

Phone 14: 30/70

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.922 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.917 (+/- 0.08)

SVM 2 Accuracy (C=20): 0.922 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.935 (+/- 0.20)

Random Forest (10) mean accuracy score: 0.878

Random Forest Score: 0.8782608695652174

Random Forest (50) mean accuracy score: 0.887

Random Forest Score: 0.8869565217391304

MLP mean accuracy mean accuracy: 0.913

MLP Score: 0.9130434782608695

VC mean accuracy: 0.904

VC Score: 0.9043478260869565

Phone 15: 80/20

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 0.952 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.798 (+/- 0.07)

SVM 2 Accuracy (C=20): 0.952 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.847 (+/- 0.29)

Random Forest (10) mean accuracy score: 0.952

Random Forest Score: 0.9523809523809523

Random Forest (50) mean accuracy score: 0.952

Random Forest Score: 0.9523809523809523

MLP mean accuracy mean accuracy: 0.952

MLP Score: 0.9523809523809523

VC mean accuracy: 0.952

VC Score: 0.9523809523809523

Attack Session Results:

SVM 1 Accuracy (C=1): 0.400 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.798 (+/- 0.12)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.847 (+/- 0.29)

Random Forest (10) mean accuracy score: 0.400

Random Forest Score: 0.4

Random Forest (50) mean accuracy score: 0.400

Random Forest Score: 0.4

MLP mean accuracy mean accuracy: 0.200

MLP Score: 0.2

VC mean accuracy: 0.200

VC Score: 0.2

Phone 15: 70/30

PCA Number of Variables: 20

Model Results:

SVM 1 Accuracy (C=1): 0.906 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.782 (+/- 0.10)

SVM 2 Accuracy (C=20): 0.938 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.845 (+/- 0.30)

Random Forest (10) mean accuracy score: 0.969

Random Forest Score: 0.96875

Random Forest (50) mean accuracy score: 0.938

Random Forest Score: 0.9375

MLP mean accuracy mean accuracy: 0.938

MLP Score: 0.9375

VC mean accuracy: 0.938

VC Score: 0.9375

Attack Session Results:

SVM 1 Accuracy (C=1): 0.400 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.754 (+/- 0.10)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.845 (+/- 0.30)

Random Forest (10) mean accuracy score: 0.800

Random Forest Score: 0.8

Random Forest (50) mean accuracy score: 0.400

Random Forest Score: 0.8

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.200

VC Score: 0.2

Phone 15: 30/70

PCA Number of Variables: 12 Model Results:

SVM 1 Accuracy (C=1): 0.595 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.775 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.757 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.800 (+/- 0.37)

Random Forest (10) mean accuracy score: 0.784

Random Forest Score: 0.7837837837837838

Random Forest (50) mean accuracy score: 0.757

Random Forest Score: 0.7567567567567568

MLP mean accuracy mean accuracy: 0.784

MLP Score: 0.7837837837837838

VC mean accuracy: 0.811

VC Score: 0.8108108108108109

Attack Session Results:

SVM 1 Accuracy (C=1): 0.200 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.719 (+/- 0.56)

SVM 2 Accuracy (C=20): 0.400 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.800 (+/- 0.37)

Random Forest (10) mean accuracy score: 0.400

Random Forest Score: 0.4

Random Forest (50) mean accuracy score: 0.400

Random Forest Score: 0.4

MLP mean accuracy mean accuracy: 0.400

MLP Score: 0.4

VC mean accuracy: 0.400

VC Score: 0.4

Phone 16: 80/20

PCA Number of Variables: 34

Model Results:

SVM 1 Accuracy (C=1): 0.818 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.908 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.818 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.929 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.909

Random Forest Score: 0.9090909090909091

Random Forest (50) mean accuracy score: 0.818

Random Forest Score: 0.8181818181818182

MLP mean accuracy mean accuracy: 0.773

MLP Score: 0.7727272727272727

VC mean accuracy: 0.864

VC Score: 0.8636363636363636

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.875 (+/- 0.20)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.929 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.200

Random Forest Score: 0.2

Random Forest (50) mean accuracy score: 0.200

Random Forest Score: 0.2

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 16: 70/30

PCA Number of Variables: 32

Model Results:

SVM 1 Accuracy (C=1): 0.758 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.935 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.758 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.971 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.879

Random Forest Score: 0.8787878787878788

Random Forest (50) mean accuracy score: 0.848

Random Forest Score: 0.8484848484848485

MLP mean accuracy mean accuracy: 0.788

MLP Score: 0.7878787878787878

VC mean accuracy: 0.788

VC Score: 0.7878787878787878

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.934 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.971 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.200

Random Forest Score: 0.2

Random Forest (50) mean accuracy score: 0.200

Random Forest Score: 0.2

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 16: 30/70

PCA Number of Variables: 19

Model Results:

SVM 1 Accuracy (C=1): 0.883 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.908 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.883 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.975 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.857

Random Forest Score: 0.8571428571428571

Random Forest (50) mean accuracy score: 0.792

Random Forest Score: 0.7922077922077922

MLP mean accuracy mean accuracy: 0.896

MLP Score: 0.8961038961038961

VC mean accuracy: 0.883

VC Score: 0.8831168831168831

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.937 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.975 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.000

Random Forest Score: 0.0

Random Forest (50) mean accuracy score: 0.000

Random Forest Score: 0.0

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 17: 80/20

PCA Number of Variables: 33

Model Results:

SVM 1 Accuracy (C=1): 0.973 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.905 (+/- 0.00)

SVM 2 Accuracy (C=20): 1.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.911 (+/- 0.14)

Random Forest (10) mean accuracy score: 0.892

Random Forest Score: 0.8918918918918919

Random Forest (50) mean accuracy score: 0.892

Random Forest Score: 0.8918918918918919

MLP mean accuracy mean accuracy: 0.973

MLP Score: 0.972972972972973

VC mean accuracy: 0.946

VC Score: 0.9459459459459459

Phone 17: 70/30

PCA Number of Variables: 30 Model Results:

SVM 1 Accuracy (C=1): 0.964 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.891 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.964 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.913 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.964

Random Forest Score: 0.9642857142857143

Random Forest (50) mean accuracy score: 0.964

Random Forest Score: 0.9642857142857143

MLP mean accuracy mean accuracy: 0.929

MLP Score: 0.9285714285714286

VC mean accuracy: 0.964

VC Score: 0.9642857142857143

Phone 17: 30/70

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 0.899 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.800 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.899 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.850 (+/- 0.24)

Random Forest (10) mean accuracy score: 0.884

Random Forest Score: 0.8837209302325582

Random Forest (50) mean accuracy score: 0.884

Random Forest Score: 0.8837209302325582

MLP mean accuracy mean accuracy: 0.915

MLP Score: 0.9147286821705426

VC mean accuracy: 0.891

VC Score: 0.8914728682170543

Phone 18: 80/20

PCA Number of Variables: 32

Model Results:

SVM 1 Accuracy (C=1): 1.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.957 (+/- 0.06)

SVM 2 Accuracy (C=20): 1.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.979 (+/- 0.05)

Random Forest (10) mean accuracy score: 1.000

Random Forest Score: 1.0

Random Forest (50) mean accuracy score: 1.000

Random Forest Score: 1.0

MLP mean accuracy mean accuracy: 1.000

MLP Score: 1.0

VC mean accuracy: 1.000

VC Score: 1.0

Phone 18: 70/30

PCA Number of Variables: 30

Model Results:

SVM 1 Accuracy (C=1): 0.971 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.938 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.986 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.964 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.986

Random Forest Score: 0.9855072463768116

Random Forest (50) mean accuracy score: 0.986

Random Forest Score: 0.9855072463768116

MLP mean accuracy mean accuracy: 0.986

MLP Score: 0.9855072463768116

VC mean accuracy: 0.986

VC Score: 0.9855072463768116

Phone 18: 30/70

PCA Number of Variables: 20

Model Results:

SVM 1 Accuracy (C=1): 0.957 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.928 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.957 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.959 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.957

Random Forest Score: 0.9565217391304348

Random Forest (50) mean accuracy score: 0.969

Random Forest Score: 0.968944099378882

MLP mean accuracy mean accuracy: 0.963

MLP Score: 0.9627329192546584

VC mean accuracy: 0.975

VC Score: 0.9751552795031055

Phone 19: 80/20

PCA Number of Variables: 18 Model Results:

SVM 1 Accuracy (C=1): 0.966 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.877 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.931 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.928 (+/- 0.12)

Random Forest (10) mean accuracy score: 0.966

Random Forest Score: 0.9655172413793104

Random Forest (50) mean accuracy score: 0.931

Random Forest Score: 0.9310344827586207

MLP mean accuracy mean accuracy: 0.897

MLP Score: 0.896551724137931

VC mean accuracy: 0.931

VC Score: 0.9310344827586207

Phone 19: 70/30

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.907 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.840 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.930 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.902 (+/- 0.19)

Random Forest (10) mean accuracy score: 0.860

Random Forest Score: 0.8604651162790697

Random Forest (50) mean accuracy score: 0.907

Random Forest Score: 0.9069767441860465

MLP mean accuracy mean accuracy: 0.860

MLP Score: 0.8604651162790697

VC mean accuracy: 0.860

VC Score: 0.8604651162790697

Phone 19: 30/70

PCA Number of Variables: 10

Model Results:

SVM 1 Accuracy (C=1): 0.861 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.762 (+/- 0.10)

SVM 2 Accuracy (C=20): 0.891 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.758 (+/- 0.30)

Random Forest (10) mean accuracy score: .891

Random Forest Score: 0.8910891089108911

Random Forest (50) mean accuracy score: 0.911

Random Forest Score: 0.9108910891089109

MLP mean accuracy mean accuracy: 0.891

MLP Score: 0.8910891089108911

VC mean accuracy: 0.881

VC Score: 0.8811881188118812

Phone 20: 80/20

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 1.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.973 (+/- 0.01)

SVM 2 Accuracy (C=20): 1.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.951 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.957

Random Forest Score: 0.9574468085106383

Random Forest (50) mean accuracy score: 0.957

Random Forest Score: 0.9574468085106383

MLP mean accuracy mean accuracy: 1.000

MLP Score: 1.0

VC mean accuracy: 0.957

VC Score: 0.9574468085106383

Phone 20: 70/30

PCA Number of Variables: 20

Model Results:

SVM 1 Accuracy (C=1): 0.971 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.975 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.971 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.981 (+/- 0.06)

Random Forest (10) mean accuracy score: 0.957

Random Forest Score: 0.9571428571428572

Random Forest (50) mean accuracy score: 0.957

Random Forest Score: 0.9571428571428572

MLP mean accuracy mean accuracy: 0.957

MLP Score: 0.9571428571428572

VC mean accuracy: 0.957

VC Score: 0.9571428571428572

Phone 20: 30/70

PCA Number of Variables: 11

Model Results:

SVM 1 Accuracy (C=1): 0.963 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.942 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.963 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.936 (+/- 0.22)

Random Forest (10) mean accuracy score: 0.932

Random Forest Score: 0.9320987654320988

Random Forest (50) mean accuracy score: 0.938

Random Forest Score: 0.9382716049382716

MLP mean accuracy mean accuracy: 0.963

MLP Score: 0.9629629629629629

VC mean accuracy: 0.951

VC Score: 0.9506172839506173

Phone 21: 80/20

PCA Number of Variables: 14

Model Results:

SVM 1 Accuracy (C=1): 0.950 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.934 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.950 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.926 (+/- 0.21)

Random Forest (10) mean accuracy score: 0.950

Random Forest Score: 0.95

Random Forest (50) mean accuracy score: 0.950

Random Forest Score: 0.95

MLP mean accuracy mean accuracy: 0.950

MLP Score: 0.95

VC mean accuracy: 0.950

VC Score: 0.95

Phone 21: 70/30

PCA Number of Variables: 13

Model Results:

SVM 1 Accuracy (C=1): 0.966 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.926 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.966 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.927 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.966

Random Forest Score: 0.9655172413793104

Random Forest (50) mean accuracy score: 0.966

Random Forest Score: 0.9655172413793104

MLP mean accuracy mean accuracy: 0.966

MLP Score: 0.9655172413793104

VC mean accuracy: 0.966

VC Score: 0.9655172413793104

Phone 21: 30/70

PCA Number of Variables: 5

Model Results:

SVM 1 Accuracy (C=1): 0.926 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.926 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.926 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.926 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.941

Random Forest Score: 0.9411764705882353

Random Forest (50) mean accuracy score: 0.941

Random Forest Score: 0.9411764705882353

MLP mean accuracy mean accuracy: 0.926

MLP Score: 0.9264705882352942

VC mean accuracy: 0.941

VC Score: 0.9411764705882353

Phone 22: 80/20

PCA Number of Variables: 45

Model Results:

SVM 1 Accuracy (C=1): 0.872 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.923 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.872 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.967 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.897

Random Forest Score: 0.8974358974358975

Random Forest (50) mean accuracy score: 0.923

Random Forest Score: 0.9230769230769231

MLP mean accuracy mean accuracy: 0.923

MLP Score: 0.9230769230769231

VC mean accuracy: 0.949

VC Score: 0.9487179487179487

Phone 22: 70/30

PCA Number of Variables: 45

Model Results:

SVM 1 Accuracy (C=1): 0.932 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.926 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.881 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.947 (+/- 0.07)

Random Forest (10) mean accuracy score: 0.898

Random Forest Score: 0.8983050847457628

Random Forest (50) mean accuracy score: 0.915

Random Forest Score: 0.9152542372881356

MLP mean accuracy mean accuracy: 0.915

MLP Score: 0.9152542372881356

VC mean accuracy: 0.932

VC Score: 0.9322033898305084

Phone 22: 30/70

PCA Number of Variables: 30

Model Results:

SVM 1 Accuracy (C=1): 0.927 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 1.000 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.927 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.963 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.883

Random Forest Score: 0.8832116788321168

Random Forest (50) mean accuracy score: 0.883

Random Forest Score: 0.8832116788321168

MLP mean accuracy mean accuracy: 0.934

MLP Score: 0.9343065693430657

VC mean accuracy: 0.942

VC Score: 0.9416058394160584

Phone 23: 80/20

PCA Number of Variables: 36

Model Results:

SVM 1 Accuracy (C=1): 0.938 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.872 (+/- 0.06)

SVM 2 Accuracy (C=20): 0.875 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.914 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.844

Random Forest Score: 0.84375

Random Forest (50) mean accuracy score: 0.844

Random Forest Score: 0.84375

MLP mean accuracy mean accuracy: 0.969

MLP Score: 0.96875

VC mean accuracy: 0.906

VC Score: 0.90625

Phone 23: 70/30

PCA Number of Variables: 34

Model Results:

SVM 1 Accuracy (C=1): 0.958 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.863 (+/- 0.09)

SVM 2 Accuracy (C=20): 0.875 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.873 (+/- 0.20)

Random Forest (10) mean accuracy score: 0.896

Random Forest Score: 0.8958333333333334

Random Forest (50) mean accuracy score: 0.958

Random Forest Score: 0.9583333333333334

MLP mean accuracy mean accuracy: 0.896

MLP Score: 0.8958333333333334

VC mean accuracy: 0.917

VC Score: 0.9166666666666666

Phone 23: 30/70

PCA Number of Variables: 22

Model Results:

SVM 1 Accuracy (C=1): 0.864 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.853 (+/- 0.21)

SVM 2 Accuracy (C=20): 0.864 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.855 (+/- 0.26)

Random Forest (10) mean accuracy score: 0.864

Random Forest Score: 0.8636363636363636

Random Forest (50) mean accuracy score: 0.864

Random Forest Score: 0.8636363636363636

MLP mean accuracy mean accuracy: 0.918

MLP Score: 0.9181818181818182

VC mean accuracy: 0.873

VC Score: 0.8727272727272727

Phone 24: 80/20

PCA Number of Variables: 33

Model Results:

SVM 1 Accuracy (C=1): 0.944 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.937 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.972 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.924 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.944

Random Forest Score: 0.9444444444444444

Random Forest (50) mean accuracy score: 0.944

Random Forest Score: 0.9444444444444444

MLP mean accuracy mean accuracy: 0.972

MLP Score: 0.9722222222222222

VC mean accuracy: 0.944

VC Score: 0.9444444444444444

Phone 24: 70/30

PCA Number of Variables: 30

Model Results:

SVM 1 Accuracy (C=1): 0.981 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.952 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.963 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.913 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.944

Random Forest Score: 0.9444444444444444

Random Forest (50) mean accuracy score: 0.963

Random Forest Score: 0.9629629629629629

MLP mean accuracy mean accuracy: 0.944

MLP Score: 0.9444444444444444

VC mean accuracy: 0.963

VC Score: 0.9629629629629629

Phone 24: 30/70

PCA Number of Variables: 16

Model Results:

SVM 1 Accuracy (C=1): 0.960 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.963 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.944 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.986 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.960

Random Forest Score: 0.9603174603174603

Random Forest (50) mean accuracy score: 0.960

Random Forest Score: 0.9603174603174603

MLP mean accuracy mean accuracy: 0.937

MLP Score: 0.9365079365079365

VC mean accuracy: 0.960

VC Score: 0.9603174603174603

Phone 25: 80/20

PCA Number of Variables: 19

Model Results:

SVM 1 Accuracy (C=1): 0.958 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.935 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.917 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.871 (+/- 0.23)

Random Forest (10) mean accuracy score: 0.958

Random Forest Score: 0.9583333333333334

Random Forest (50) mean accuracy score: 0.958

Random Forest Score: 0.9583333333333334

MLP mean accuracy mean accuracy: 0.917

MLP Score: 0.9166666666666666

VC mean accuracy: 0.958

VC Score: 0.9583333333333334

Phone 25: 70/30

PCA Number of Variables: 18

Model Results:

SVM 1 Accuracy (C=1): 0.972 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.926 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.944 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.922 (+/- 0.26)

Random Forest (10) mean accuracy score: 0.972

Random Forest Score: 0.9722222222222222

Random Forest (50) mean accuracy score: 0.972

Random Forest Score: 0.9722222222222222

MLP mean accuracy mean accuracy: 0.944

MLP Score: 0.9444444444444444

VC mean accuracy: 0.972

VC Score: 0.9722222222222222

Phone 25: 30/70

PCA Number of Variables:10

Model Results: SVM 1 Accuracy (C=1): 0.963 (+/- 0.00) SVM 1 Accuracy cross validated (cv=2): 0.963 (+/- 0.02) SVM 2 Accuracy (C=20): 0.963 (+/- 0.00) SVM 2 Accuracy cross validated (cv=20): 0.953 (+/- 0.12) Random Forest (10) mean accuracy score: 0.976 Random Forest Score: 0.975609756097561 Random Forest (50) mean accuracy score: 0.963 Random Forest Score: 0.9634146341463414 MLP mean accuracy mean accuracy: 0.963 MLP Score: 0.9634146341463414 VC mean accuracy: 0.963 VC Score: 0.9634146341463414

Phone 26: 80/20

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.933 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.948 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.867 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.920 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.900

Random Forest Score: 0.9

Random Forest (50) mean accuracy score: 0.933

Random Forest Score: 0.9333333333333333

MLP mean accuracy mean accuracy: 0.867

MLP Score: 0.8666666666666667

VC mean accuracy: 0.900

VC Score: 0.9

Phone 26: 70/30

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.932 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.951 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.932 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.930 (+/- 0.13)

Random Forest (10) mean accuracy score: 0.955

Random Forest Score: 0.9545454545454546

Random Forest (50) mean accuracy score: 0.955

Random Forest Score: 0.9545454545454546

MLP mean accuracy mean accuracy: 0.955

MLP Score: 0.9545454545454546

VC mean accuracy: 0.932

VC Score: 0.9318181818181818

Phone 26: 30/70

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.883 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.953 (+/- 0.00)

SVM 2 Accuracy (C=20): 0.883 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.905 (+/- 0.23)

Random Forest (10) mean accuracy score: 0.816

Random Forest Score: 0.8155339805825242

Random Forest (50) mean accuracy score: 0.883

Random Forest Score: 0.883495145631068

MLP mean accuracy mean accuracy: 0.903

MLP Score: 0.9029126213592233

VC mean accuracy: 0.864

VC Score: 0.8640776699029126

Phone 27: 80/20

PCA Number of Variables: 31

Model Results:

SVM 1 Accuracy (C=1): 0.880 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.910 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.920 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.851 (+/- 0.17)

Random Forest (10) mean accuracy score: 0.920

Random Forest Score: 0.92

Random Forest (50) mean accuracy score: 0.880

Random Forest Score: 0.88

MLP mean accuracy mean accuracy: 0.960

MLP Score: 0.96

VC mean accuracy: 0.960

VC Score: 0.96

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.860 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.851 (+/- 0.17)

Random Forest (10) mean accuracy score: 0.400

Random Forest Score: 0.4

Random Forest (50) mean accuracy score: 0.000

Random Forest Score: 0.4

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 27: 70/30

PCA Number of Variables: 30

Model Results:

SVM 1 Accuracy (C=1): 0.895 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.896 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.789 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.874 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.895

Random Forest Score: 0.8947368421052632

Random Forest (50) mean accuracy score: 0.947

Random Forest Score: 0.9473684210526315

MLP mean accuracy mean accuracy: 0.947

MLP Score: 0.9473684210526315

VC mean accuracy: 0.895

VC Score: 0.8947368421052632

Attack Session Results: SVM 1 Accuracy (C=1): 0.895 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.896 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.789 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.874 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.895

Random Forest Score: 0.8947368421052632

Random Forest (50) mean accuracy score: 0.947

Random Forest Score: 0.9473684210526315

MLP mean accuracy mean accuracy: 0.947

MLP Score: 0.9473684210526315

VC mean accuracy: 0.895

VC Score: 0.8947368421052632

Phone 27: 30/70

PCA Number of Variables: 16 Model Results:

SVM 1 Accuracy (C=1): 0.932 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.727 (+/- 0.23)

SVM 2 Accuracy (C=20): 0.864 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.758 (+/- 0.42)

Random Forest (10) mean accuracy score: 0.898

Random Forest Score: 0.8977272727272727

Random Forest (50) mean accuracy score: 0.909

Random Forest Score: 0.9090909090909091

MLP mean accuracy mean accuracy: 0.977

MLP Score: 0.9772727272727273

VC mean accuracy: 0.966

VC Score: 0.9659090909090909

Attack Session Results:

SVM 1 Accuracy (C=1): 0.200 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.811 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.758 (+/- 0.42)

Random Forest (10) mean accuracy score: 0.200

Random Forest Score: 0.2

Random Forest (50) mean accuracy score: 0.200

Random Forest Score: 0.2

MLP mean accuracy mean accuracy: 0.200

MLP Score: 0.2

VC mean accuracy: 0.200

VC Score: 0.2

Phone 28: 80/20

PCA Number of Variables: 48 Model Results:

SVM 1 Accuracy (C=1): 0.881 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.898 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.905 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.912 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.929

Random Forest Score: 0.9285714285714286

Random Forest (50) mean accuracy score: 0.905

Random Forest Score: 0.9047619047619048

MLP mean accuracy mean accuracy: 0.905

MLP Score: 0.9047619047619048

VC mean accuracy: 0.952

VC Score: 0.9523809523809523

Phone 28: 70/30

PCA Number of Variables: 45

Model Results:

SVM 1 Accuracy (C=1): 0.905 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.890 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.905 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.897 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.889

Random Forest Score: 0.8888888888888888

Random Forest (50) mean accuracy score: 0.921

Random Forest Score: 0.9206349206349206

MLP mean accuracy mean accuracy: 0.952

MLP Score: 0.9523809523809523

VC mean accuracy: 0.937

VC Score: 0.9365079365079365

Phone 28: 30/70

PCA Number of Variables: 29

Model Results:

SVM 1 Accuracy (C=1): 0.878 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.872 (+/- 0.06)

SVM 2 Accuracy (C=20): 0.878 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.881 (+/- 0.27)

Random Forest (10) mean accuracy score: 0.844

Random Forest Score: 0.8435374149659864

Random Forest (50) mean accuracy score: 0.850

Random Forest Score: 0.8503401360544217

MLP mean accuracy mean accuracy: 0.939

MLP Score: 0.9387755102040817

VC mean accuracy: 0.891

VC Score: 0.891156462585034

Phone 29: 80/20

PCA Number of Variables: 34

Model Results:

SVM 1 Accuracy (C=1): 0.906 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.929 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.906 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.922 (+/- 0.20)

Random Forest (10) mean accuracy score: 0.812

Random Forest Score: 0.8125

Random Forest (50) mean accuracy score: 0.844

Random Forest Score: 0.84375

MLP mean accuracy mean accuracy: 0.875

MLP Score: 0.875

VC mean accuracy: 0.875

VC Score: 0.875

Phone 29: 70/30

PCA Number of Variables: 35

Model Results:

SVM 1 Accuracy (C=1): 0.917 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.955 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.938 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.874 (+/- 0.26)

Random Forest (10) mean accuracy score: 0.854

Random Forest Score: 0.8541666666666666

Random Forest (50) mean accuracy score: 0.875

Random Forest Score: 0.875

MLP mean accuracy mean accuracy: 0.917

MLP Score: 0.9166666666666666

VC mean accuracy: 0.896

VC Score: 0.8958333333333334

Phone 29: 30/70

PCA Number of Variables: 19

Model Results:

SVM 1 Accuracy (C=1): 0.848 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.893 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.777 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.858 (+/- 0.19)

Random Forest (10) mean accuracy score: 0.857

Random Forest Score: 0.8571428571428571

Random Forest (50) mean accuracy score: 0.848

Random Forest Score: 0.8482142857142857

MLP mean accuracy mean accuracy: 0.848

MLP Score: 0.8482142857142857

VC mean accuracy: 0.839

VC Score: 0.8392857142857143

Phone 30: 80/20

PCA Number of Variables: 28

Model Results:

SVM 1 Accuracy (C=1): 0.880 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.938 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.880 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.814 (+/- 0.18)

Random Forest (10) mean accuracy score: 0.960

Random Forest Score: 0.96

Random Forest (50) mean accuracy score: 0.960

Random Forest Score: 0.96

MLP mean accuracy mean accuracy: 0.960

MLP Score: 0.96

VC mean accuracy: 0.960

VC Score: 0.96

Phone 30: 70/30

PCA Number of Variables: 27

Model Results:

SVM 1 Accuracy (C=1): 0.892 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.941 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.919 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.814 (+/- 0.31)

Random Forest (10) mean accuracy score: 0.973

Random Forest Score: 0.972972972972973

Random Forest (50) mean accuracy score: 0.946

Random Forest Score: 0.9459459459459459

MLP mean accuracy mean accuracy: 0.973

MLP Score: 0.972972972972973

VC mean accuracy: 0.973

VC Score: 0.972972972972973

Phone 30: 30/70

PCA Number of Variables: 18

Model Results:

SVM 1 Accuracy (C=1): 0.884 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.862 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.884 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.922 (+/- 0.25)

Random Forest (10) mean accuracy score: 0.919

Random Forest Score: 0.9186046511627907

Random Forest (50) mean accuracy score: 0.895

Random Forest Score: 0.8953488372093024

MLP mean accuracy mean accuracy: 0.953

MLP Score: 0.9534883720930233

VC mean accuracy: 0.930

VC Score: 0.9302325581395349

Phone 31: 80/20

PCA Number of Variables: 20

Model Results:

SVM 1 Accuracy (C=1): 0.800 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.874 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.800 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.839 (+/- 0.22)

Random Forest (10) mean accuracy score: 0.867

Random Forest Score: 0.8666666666666667

Random Forest (50) mean accuracy score: 0.767

Random Forest Score: 0.7666666666666667

MLP mean accuracy mean accuracy: 0.867

MLP Score: 0.8666666666666667

VC mean accuracy: 0.900

VC Score: 0.9

Phone 31: 70/30

PCA Number of Variables: 33

Model Results:

SVM 1 Accuracy (C=1): 0.933 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.856 (+/- 0.10)

SVM 2 Accuracy (C=20): 0.889 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.834 (+/- 0.19)

Random Forest (10) mean accuracy score: 0.911

Random Forest Score: 0.9111111111111111

Random Forest (50) mean accuracy score: 0.867

Random Forest Score: 0.8666666666666667

MLP mean accuracy mean accuracy: 0.911

MLP Score: 0.9111111111111111

VC mean accuracy: 0.911

VC Score: 0.9111111111111111

Phone 31: 30/70

PCA Number of Variables: 19

Model Results:

SVM 1 Accuracy (C=1): 0.838

SVM 1 Accuracy cross validated (cv=2): 0.701 (+/- 0.16)

SVM 2 Accuracy (C=20): 0.848 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.702 (+/- 0.42)

Random Forest (10) mean accuracy score: 0.790

Random Forest Score: 0.7904761904761904

Random Forest (50) mean accuracy score: 0.810

Random Forest Score: 0.8095238095238095

MLP mean accuracy mean accuracy: 0.886

MLP Score: 0.8857142857142857

VC mean accuracy: 0.857

VC Score: 0.8571428571428571

Phone 32: 80/20

PCA Number of Variables: 40

Model Results:

SVM 1 Accuracy (C=1): 0.944 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.917 (+/- 0.08)

SVM 2 Accuracy (C=20): 0.944 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.931 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.833

Random Forest Score: 0.8333333333333334

Random Forest (50) mean accuracy score: 0.889

Random Forest Score: 0.8888888888888888

MLP mean accuracy mean accuracy: 0.944

MLP Score: 0.9444444444444444

VC mean accuracy: 0.972

VC Score: 0.9722222222222222

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.952 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.931 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.400

Random Forest Score: 0.4

Random Forest (50) mean accuracy score: 0.200

Random Forest Score: 0.4

MLP mean accuracy mean accuracy: 0.200

MLP Score: 0.2

VC mean accuracy: 0.200

VC Score: 0.2

Phone 32: 70/30

PCA Number of Variables: 40

Model Results:

SVM 1 Accuracy (C=1): 0.870 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.929 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.870 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.928 (+/- 0.14)

Random Forest (10) mean accuracy score: 0.852

Random Forest Score: 0.8518518518518519

Random Forest (50) mean accuracy score: 0.889

Random Forest Score: 0.8888888888888888

MLP mean accuracy mean accuracy: 0.907

MLP Score: 0.9074074074074074

VC mean accuracy: 0.889

VC Score: 0.8888888888888888

Attack Session Results:

SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.937 (+/- 0.03)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.928 (+/- 0.14)

Random Forest (10) mean accuracy score: 0.400

Random Forest Score: 0.4

Random Forest (50) mean accuracy score: 0.600

Random Forest Score: 0.4

MLP mean accuracy mean accuracy: 0.200

MLP Score: 0.2

VC mean accuracy: 0.400

VC Score: 0.4

Phone 32: 30/70

PCA Number of Variables: 22

Model Results:

SVM 1 Accuracy (C=1): 0.905 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.944 (+/- 0.11)

SVM 2 Accuracy (C=20): 0.905 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.960 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.881

Random Forest Score: 0.8809523809523809

Random Forest (50) mean accuracy score: 0.905

Random Forest Score: 0.9047619047619048

MLP mean accuracy mean accuracy: 0.944

MLP Score: 0.9444444444444444

VC mean accuracy: 0.921

VC Score: 0.9206349206349206

Attack Session Results: SVM 1 Accuracy (C=1): 0.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.963 (+/- 0.07)

SVM 2 Accuracy (C=20): 0.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=10): 0.960 (+/- 0.16)

Random Forest (10) mean accuracy score: 0.400

Random Forest Score: 0.4

Random Forest (50) mean accuracy score: 0.200

Random Forest Score: 0.4

MLP mean accuracy mean accuracy: 0.000

MLP Score: 0.0

VC mean accuracy: 0.000

VC Score: 0.0

Phone 33: 80/20

PCA Number of Variables: 25

Model Results:

SVM 1 Accuracy (C=1): 1.000 (+/- 0.00) SVM 1 Accuracy cross validated (cv=2): 0.936 (+/- 0.02) SVM 2 Accuracy (C=20): 0.964 (+/- 0.00) SVM 2 Accuracy cross validated (cv=20): 0.945 (+/- 0.15) Random Forest (10) mean accuracy score: 0.964 Random Forest Score: 0.9642857142857143 Random Forest (50) mean accuracy score: 0.964 Random Forest Score: 0.9642857142857143 MLP mean accuracy mean accuracy: 0.893 MLP Score: 0.8928571428571429 VC mean accuracy: 0.929 VC Score: 0.9285714285714286

Phone 33: 70/30

PCA Number of Variables: 22

Model Results:

SVM 1 Accuracy (C=1): 0.952 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.969 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.952 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.947 (+/- 0.15)

Random Forest (10) mean accuracy score: 0.952

Random Forest Score: 0.9523809523809523

Random Forest (50) mean accuracy score: 0.929

Random Forest Score: 0.9285714285714286

MLP mean accuracy mean accuracy: 0.976

MLP Score: 0.9761904761904762

VC mean accuracy: 0.952

VC Score: 0.9523809523809523

Phone 33: 30/70

PCA Number of Variables: 19

Model Results:

SVM 1 Accuracy (C=1): 0.959 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.804 (+/- 0.11)

SVM 2 Accuracy (C=20): 0.928 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.830 (+/- 0.42)

Random Forest (10) mean accuracy score: 0.969

Random Forest Score: 0.9690721649484536

Random Forest (50) mean accuracy score: 0.979

Random Forest Score: 0.979381443298969

MLP mean accuracy mean accuracy: 0.928

MLP Score: 0.9278350515463918

VC mean accuracy: 0.969

VC Score: 0.9690721649484536

Phone 34: 80/20

PCA Number of Variables: 21

Model Results:

SVM 1 Accuracy (C=1): 0.969 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.896 (+/- 0.11)

SVM 2 Accuracy (C=20): 0.906 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.860 (+/- 0.22)

Random Forest (10) mean accuracy score: 1.000

Random Forest Score: 1.0

Random Forest (50) mean accuracy score: 0.969

Random Forest Score: 0.96875

MLP mean accuracy mean accuracy: 1.000

MLP Score: 1.0

VC mean accuracy: 0.969

VC Score: 0.96875

Phone 34: 70/30

PCA Number of Variables: 19

Model Results:

SVM 1 Accuracy (C=1): 1.000 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.863 (+/- 0.05)

SVM 2 Accuracy (C=20): 0.896 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.858 (+/- 0.22)

Random Forest (10) mean accuracy score: 0.979

Random Forest Score: 0.9791666666666666

Random Forest (50) mean accuracy score: 0.979

Random Forest Score: 0.9791666666666666

MLP mean accuracy mean accuracy: 0.979

MLP Score: 0.9791666666666666

VC mean accuracy: 0.979

VC Score: 0.9791666666666666

Phone 34: 30/70

PCA Number of Variables: 11

Model Results:

SVM 1 Accuracy (C=1): 0.873 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.852 (+/- 0.12)

SVM 2 Accuracy (C=20): 0.873 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.983 (+/- 0.10)

Random Forest (10) mean accuracy score: 0.918

Random Forest Score: 0.9181818181818182

Random Forest (50) mean accuracy score: 0.927

Random Forest Score: 0.9272727272727272

MLP mean accuracy mean accuracy: 0.891

MLP Score: 0.8909090909090909

VC mean accuracy: 0.936

VC Score: 0.9363636363636364

Phone 35: 80/20

PCA Number of Variables: 43

Model Results:

SVM 1 Accuracy (C=1): 0.981 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.944 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.963 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.940 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.963

Random Forest Score: 0.9629629629629629

Random Forest (50) mean accuracy score: 0.981

Random Forest Score: 0.9814814814814815

MLP mean accuracy mean accuracy: 0.963

MLP Score: 0.9629629629629629

VC mean accuracy: 0.981

VC Score: 0.9814814814814815

Phone 35: 70/30

PCA Number of Variables: 40 Model Results:

SVM 1 Accuracy (C=1): 0.938 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.963 (+/- 0.01)

SVM 2 Accuracy (C=20): 0.951 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.927 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.963

Random Forest Score: 0.9629629629629629

Random Forest (50) mean accuracy score: 0.975

Random Forest Score: 0.9753086419753086

MLP mean accuracy mean accuracy: 0.975

MLP Score: 0.9753086419753086

VC mean accuracy: 0.975

VC Score: 0.9753086419753086

Phone 35: 30/70

PCA Number of Variables: 25

Model Results:

SVM 1 Accuracy (C=1): 0.968 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.963 (+/- 0.02)

SVM 2 Accuracy (C=20): 0.937 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.872 (+/- 0.29)

Random Forest (10) mean accuracy score: 0.968

Random Forest Score: 0.9682539682539683

Random Forest (50) mean accuracy score: 0.968

Random Forest Score: 0.9682539682539683

MLP mean accuracy mean accuracy: 0.958

MLP Score: 0.9576719576719577

VC mean accuracy: 0.968

VC Score: 0.9682539682539683

Phone 36: 80/20

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.967 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.914 (+/- 0.00)

SVM 2 Accuracy (C=20): 1.000 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.958 (+/- 0.11)

Random Forest (10) mean accuracy score: 0.967

Random Forest Score: 0.9666666666666667

Random Forest (50) mean accuracy score: 1.000

Random Forest Score: 1.0

MLP mean accuracy mean accuracy: 1.000

MLP Score: 1.0

VC mean accuracy: 1.000

VC Score: 1.0

Phone 36: 70/30

PCA Number of Variables: 17

Model Results:

SVM 1 Accuracy (C=1): 0.977 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.902 (+/- 0.04)

SVM 2 Accuracy (C=20): 0.955 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.932 (+/- 0.09)

Random Forest (10) mean accuracy score: 0.977

Random Forest Score: 0.9772727272727273

Random Forest (50) mean accuracy score: 0.977

Random Forest Score: 0.9772727272727273

MLP mean accuracy mean accuracy: 0.955

MLP Score: 0.9545454545454546

VC mean accuracy: 1.000

VC Score: 1.0

Phone 36: 30/70

PCA Number of Variables: 11

Model Results:

SVM 1 Accuracy (C=1): 0.951 (+/- 0.00)

SVM 1 Accuracy cross validated (cv=2): 0.839 (+/- 0.13)

SVM 2 Accuracy (C=20): 0.942 (+/- 0.00)

SVM 2 Accuracy cross validated (cv=20): 0.875 (+/- 0.27)

Random Forest (10) mean accuracy score: 0.961

Random Forest Score: 0.9611650485436893

Random Forest (50) mean accuracy score: 0.981

Random Forest Score: 0.9805825242718447

MLP mean accuracy mean accuracy: 0.922

MLP Score: 0.9223300970873787

VC mean accuracy: 0.971

VC Score: 0.970873786407767

G. ONE-CLASS STATISTICS FOR INDIVIDUAL PHONES

Phone 1: 80/20

Linear test success rate: 2/2

1.0

RBF test success rate: 1/2

0.5

Poly test success rate: 0/2

0.0

Sig test success rate: 2/2

1.0

Phone 1: 70/30

Linear test success rate: 1/3

0.3333333333333333

RBF test success rate: 1/3

0.3333333333333333

Poly test success rate: 1/3

0.3333333333333333

Sig test success rate: 2/3

0.6666666666666666

Phone 1: 30/70

Linear test success rate: 0/7

0.0

RBF test success rate: 0/7

0.0

Poly test success rate: 2/7

0.2857142857142857

Sig test success rate: 2/7

0.2857142857142857

Phone 3: 80/20

Linear test success rate: 13/25

0.52

RBF test success rate: 13/25

0.52

Poly test success rate: 1/25

0.04

Sig test success rate: 21/25

0.84

Phone 3: 70/30

Linear test success rate: 22/38

0.5789473684210527

RBF test success rate: 22/38

0.5789473684210527

Poly test success rate: 1/38

0.02631578947368421

Sig test success rate: 29/38

0.7631578947368421

Phone 3: 30/70

Linear test success rate: 70/87

0.8045977011494253

RBF test success rate: 45/87

0.5172413793103449

Poly test success rate: 15/87

0.1724137931034483

Sig test success rate: 78/87

0.896551724137931

Phone 4: 80/20

Linear test success rate: 0/3

0.0

RBF test success rate: 1/3

0.3333333333333333

Poly test success rate: 1/3

0.3333333333333333

Sig test success rate: 3/3

1.0

Attack Session Results:

Linear SVM predictions: [-1 -1 -1 -1 1]

Linear EXP success rate: 1/5

0.2

RBF SVM predictions: [-1 -1 -1 -1 -1]

RBF EXP success rate: 0/5

0.0

Poly SVM predictions: [1 -1 1 1 1]

Poly EXP success rate: 4/5

0.8

Sig SVM predictions: [1 1 1 1 1]

Sig EXP success rate: 5/5

1.0

Phone 4: 70/30

Linear test success rate: 1/4

0.25

RBF test success rate: 1/4

0.25

Poly test success rate: 1/4

0.25

Sig test success rate: 3/4

0.75

Attack Session Results:

Linear SVM predictions: [1 -1 1 1 -1]

Linear EXP success rate: 3/5

0.6

RBF SVM predictions: [-1 -1 -1 -1 -1]

RBF EXP success rate: 0/5

0.0

Poly SVM predictions: [-1 -1 -1 -1 -1]

Poly EXP success rate: 0/5

0.0

Sig SVM predictions: [1 1 1 1 1]

Sig EXP success rate: 5/5

1.0

Phone 4: 30/70

Linear test success rate: 0/8

0.0

RBF test success rate: 0/8

0.0

Poly test success rate: 2/8

0.25

Sig test success rate: 6/8

0.75

Attack Session Results:

Linear SVM predictions: [-1 1 -1 -1 1]

Linear EXP success rate: 2/5

0.4

RBF SVM predictions: [-1 -1 -1 -1 -1]

RBF EXP success rate: 0/5

0.0

Poly SVM predictions: [-1 -1 1 -1 -1]

Poly EXP success rate: 1/5

0.2

Sig SVM predictions: [1 1 1 1 1]

Sig EXP success rate: 5/5

1.0

Phone 5: 80/20

Linear test success rate: 3/5

0.6

RBF test success rate: 2/5

0.4

Poly test success rate: 1/5

0.2

Sig test success rate: 5/5

1.0

Phone 5: 70/30

Linear test success rate: 0/7

0.0

RBF test success rate: 4/7

0.5714285714285714

Poly test success rate: 0/7

0.0

Sig test success rate: 4/7

0.5714285714285714

Phone 5: 30/70

Linear test success rate: 8/16

0.5

RBF test success rate: 5/16

0.3125

Poly test success rate: 1/16

0.0625

Sig test success rate: 13/16

0.8125

Phone 6: 80/20

Linear test success rate: 5/5

1.0

RBF test success rate: 0/5

0.0

Poly test success rate: 1/5

0.2

Sig test success rate: 4/5

0.8

Phone 6: 70/30

Linear test success rate: 4/7

0.5714285714285714

RBF test success rate: 1/7

0.14285714285714285

Poly test success rate: 1/7

0.14285714285714285

Sig test success rate: 5/7

0.7142857142857143

Phone 6: 30/70

Linear test success rate: 4/15

0.26666666666666666

RBF test success rate: 4/15

0.26666666666666666

Poly test success rate: 3/15

0.2

Sig test success rate: 13/15

0.8666666666666667

Phone 7: 80/20

Linear test success rate: 3/9

0.3333333333333333

RBF test success rate: 4/9

0.4444444444444444

Poly test success rate: 2/9

0.2222222222222222

Sig test success rate: 5/9

0.5555555555555556

Phone 7: 70/30

Linear test success rate: 7/14

0.5

RBF test success rate: 7/14

0.5

Poly test success rate: 2/14

0.14285714285714285

Sig test success rate: 6/14

0.42857142857142855

Phone 7: 30/70

Linear test success rate: 16/31

0.5161290322580645

RBF test success rate: 5/31

0.16129032258064516

Poly test success rate: 5/31

0.16129032258064516

Sig test success rate: 22/31

0.7096774193548387

Phone 8: 80/20

Linear test success rate: 5/9

0.5555555555555556

RBF test success rate: 6/9

0.6666666666666666

Poly test success rate: 0/9

0.0

Sig test success rate: 4/9

0.4444444444444444

Phone 8: 70/30

Linear test success rate: 6/13

0.46153846153846156

RBF test success rate: 6/13

0.46153846153846156

Poly test success rate: 2/13

0.15384615384615385

Sig test success rate: 7/13

0.5384615384615384

Phone 8: 30/70

Linear test success rate: 16/29

0.5517241379310345

RBF test success rate: 9/29

0.3103448275862069

Poly test success rate: 2/29

0.06896551724137931

Sig test success rate: 28/29

0.9655172413793104

Phone 9: 80/20

Linear test success rate: 5/10

0.5

RBF test success rate: 4/10

0.4

Poly test success rate: 2/10

0.2

Sig test success rate: 6/10

0.6

Phone 9: 70/30

Linear test success rate: 12/15

0.8

RBF test success rate: 3/15

0.2

Poly test success rate: 0/15

0.0

Sig test success rate: 12/15

0.8

Phone 9: 30/70

Linear test success rate: 24/35

0.6857142857142857

RBF test success rate: 4/35

0.11428571428571428

Poly test success rate: 2/35

0.05714285714285714

Sig test success rate: 26/35

0.7428571428571429

Phone 10: 80/20

Linear test success rate: 8/11

0.7272727272727273

RBF test success rate: 5/11

0.45454545454545453

Poly test success rate: 3/11

0.2727272727272727

Sig test success rate: 7/11

0.6363636363636364

Phone 10: 70/30

Linear test success rate: 9/17

0.5294117647058824

RBF test success rate: 7/17

0.4117647058823529

Poly test success rate: 7/17

0.4117647058823529

Sig test success rate: 11/17

0.6470588235294118

Phone 10: 30/70

Linear test success rate: 20/39

0.5128205128205128

RBF test success rate: 11/39

0.28205128205128205

Poly test success rate: 7/39

0.1794871794871795

Sig test success rate: 25/39

0.6410256410256411

Phone 11: 80/20

Linear test success rate: 0/4

0.0

RBF test success rate: 2/4

0.5

Poly test success rate: 0/4

0.0

Sig test success rate: 2/4

0.5

Phone 11: 70/30

Linear test success rate: 6/6

1.0

RBF test success rate: 3/6

0.5

Poly test success rate: 0/6

0.0

Sig test success rate: 2/6

0.3333333333333333

Phone 11: 30/70

Linear test success rate: 5/14

0.35714285714285715

RBF test success rate: 0/14

0.0

Poly test success rate: 1/14

0.07142857142857142

Sig test success rate: 9/14

0.6428571428571429

Phone 12: 80/20

Linear test success rate: 3/8

0.375

RBF test success rate: 7/8

0.875

Poly test success rate: 1/8

0.125

Sig test success rate: 6/8

0.75

Phone 12: 70/30

Linear test success rate: 4/12

0.3333333333333333

RBF test success rate: 7/12

0.5833333333333334

Poly test success rate: 2/12

0.16666666666666666

Sig test success rate: 9/12

0.75

Phone 12: 30/70

Linear test success rate: 16/26

0.6153846153846154

RBF test success rate: 7/26

0.2692307692307692

Poly test success rate: 4/26

0.15384615384615385

Sig test success rate: 14/26

0.5384615384615384

Phone 13: 80/20

Linear test success rate: 4/7

0.5714285714285714

RBF test success rate: 4/7

0.5714285714285714

Poly test success rate: 0/7

0.0

Sig test success rate: 6/7

0.8571428571428571

Phone 13: 70/30

Linear test success rate: 4/10

0.4

RBF test success rate: 6/10

0.6

Poly test success rate: 0/10

0.0

Sig test success rate: 5/10

0.5

Phone 13: 30/70

Linear test success rate: 9/23

0.391304347826087

RBF test success rate: 6/23

0.2608695652173913

Poly test success rate: 0/23

0.0

Sig test success rate: 20/23

0.8695652173913043

Phone 14: 80/20

Linear test success rate: 8/10

0.8

RBF test success rate: 3/10

0.3

Poly test success rate: 3/10

0.3

Sig test success rate: 10/10

1.0

Phone 14: 70/30

Linear test success rate: 4/15

0.26666666666666666

RBF test success rate: 5/15

0.3333333333333333

Poly test success rate: 2/15

0.13333333333333333

Sig test success rate: 14/15

0.9333333333333333

Phone 14: 30/70

Linear test success rate: 11/33

0.3333333333333333

RBF test success rate: 3/33

0.09090909090909091

Poly test success rate: 3/33

0.09090909090909091

Sig test success rate: 29/33

0.8787878787878788

Phone 15: 80/20

Linear test success rate: 5/6

0.8333333333333334

RBF test success rate: 2/6

0.3333333333333333

Poly test success rate: 2/6

0.3333333333333333

Sig test success rate: 5/6

0.8333333333333334

Attack Session Results:

Linear SVM predictions: [1 1 -1 1 1]

Linear EXP success rate: 4/5

0.8

RBF SVM predictions: [-1 -1 1 -1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [1 1 1 1 -1]

Poly EXP success rate: 4/5

0.8

Sig SVM predictions: [1 -1 1 -1 1]

Sig EXP success rate: 3/5

0.6

Phone 15: 70/30

Linear test success rate: 7/9

0.7777777777777778

RBF test success rate: 3/9

0.3333333333333333

Poly test success rate: 0/9

0.0

Sig test success rate: 8/9

0.8888888888888888

Attack Session Results:

Linear SVM predictions: [1 1 -1 1 1]

Linear EXP success rate: 4/5

0.8

RBF SVM predictions: [-1 -1 1 -1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [-1 1 -1 -1 -1]

Poly EXP success rate: 1/5

0.2

Sig SVM predictions: [-1 -1 1 -1 1]

Sig EXP success rate: 2/5

0.4

Phone 15: 30/70

Linear test success rate: 14/21

0.6666666666666666

RBF test success rate: 6/21

0.2857142857142857

Poly test success rate: 6/21

0.2857142857142857

Sig test success rate: 19/21

0.9047619047619048

Attack Session Results:

Linear SVM predictions: [-1 1 -1 1 1]

Linear EXP success rate: 3/5

0.6

RBF SVM predictions: [-1 -1 -1 -1 -1]

RBF EXP success rate: 0/5

0.0

Poly SVM predictions: [1 -1 -1 1 -1]

Poly EXP success rate: 2/5

0.4

Sig SVM predictions: [1 1 1 -1 1]

Sig EXP success rate: 4/5

0.8

Phone 16: 80/20

Linear test success rate: 3/8

0.375

RBF test success rate: 4/8

0.5

Poly test success rate: 4/8

0.5

Sig test success rate: 4/8

0.5

Attack Session Results: Linear SVM predictions: [-1 -1 1 1 1]

Linear EXP success rate: 3/5

0.6

RBF SVM predictions: [1 1 1 -1 -1]

RBF EXP success rate: 3/5

0.6

Poly SVM predictions: [-1 -1 -1 -1 -1]

Poly EXP success rate: 0/5

0.0

Sig SVM predictions: [1 1 -1 1 1]

Sig EXP success rate: 4/5

0.8

Phone 16: 70/30

Linear test success rate: 4/12

0.3333333333333333

RBF test success rate: 4/12

0.3333333333333333

Poly test success rate: 7/12

0.5833333333333334

Sig test success rate: 4/12

0.3333333333333333

Attack Session Results:

Linear SVM predictions: [-1 -1 1 1 -1]

Linear EXP success rate: 2/5

0.4

RBF SVM predictions: [1 -1 -1 -1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [-1 -1 -1 -1 -1]

Poly EXP success rate: 0/5

0.0

Sig SVM predictions: [1 1 -1 1 -1]

Sig EXP success rate: 3/5

0.6

Phone 16: 30/70

Linear test success rate: 16/28

0.5714285714285714

RBF test success rate: 0/28

0.0

Poly test success rate: 3/28

0.10714285714285714

Sig test success rate: 24/28

0.8571428571428571

Attack Session Results:

Linear SVM predictions: [1 1 -1 -1 1]

Linear EXP success rate: 3/5

0.6

RBF SVM predictions: [-1 -1 -1 -1 -1]

RBF EXP success rate: 0/5

0.0

Poly SVM predictions: [-1 -1 1 -1 -1]

Poly EXP success rate: 1/5

0.2

Sig SVM predictions: [-1 1 -1 1 1]

Sig EXP success rate: 3/5

0.6

Phone 17: 80/20

Linear test success rate: 4/6

0.6666666666666666

RBF test success rate: 2/6

0.3333333333333333

Poly test success rate: 1/6

0.16666666666666666

Sig test success rate: 5/6

0.8333333333333334

Phone 17: 70/30

Linear test success rate: 1/9

0.1111111111111111

RBF test success rate: 4/9

0.4444444444444444

Poly test success rate: 2/9

0.2222222222222222

Sig test success rate: 5/9

0.5555555555555556

Phone 17: 30/70

Linear test success rate: 12/20

0.6

RBF test success rate: 0/20

0.0

Poly test success rate: 2/20

0.1

Sig test success rate: 19/20

0.95

Phone 18: 80/20

Linear test success rate: 0/4

0.0

RBF test success rate: 2/4

0.5

Poly test success rate: 0/4

0.0

Sig test success rate: 1/4

0.25

Phone 18: 70/30

Linear test success rate: 5/6

0.8333333333333334

RBF test success rate: 1/6

0.16666666666666666

Poly test success rate: 3/6

0.5

Sig test success rate: 2/6

0.3333333333333333

Phone 18: 30/70

Linear test success rate: 9/13

0.6923076923076923

RBF test success rate: 0/13

0.0

Poly test success rate: 6/13

0.46153846153846156

Sig test success rate: 6/13

0.46153846153846156

Phone 19: 80/20

Linear test success rate: 4/5

0.8

RBF test success rate: 1/5

0.2

Poly test success rate: 2/5

0.4

Sig test success rate: 3/5

0.6

Phone 19: 70/30

Linear test success rate: 1/7

0.14285714285714285

RBF test success rate: 2/7

0.2857142857142857

Poly test success rate: 2/7

0.2857142857142857

Sig test success rate: 4/7

0.5714285714285714

Phone 19: 30/70

Linear test success rate: 5/15

0.3333333333333333

RBF test success rate: 1/15

0.06666666666666667

Poly test success rate: 4/15

0.26666666666666666

Sig test success rate: 11/15

0.7333333333333333

Phone 20: 80/20

Linear test success rate: 3/4

0.75

RBF test success rate: 2/4

0.5

Poly test success rate: 0/4

0.0

Sig test success rate: 3/4

0.75

Phone 20: 70/30

Linear test success rate: 2/6

0.3333333333333333

RBF test success rate: 2/6

0.3333333333333333

Poly test success rate: 1/6

0.16666666666666666

Sig test success rate: 3/6

0.5

Phone 20: 30/70

Linear test success rate: 7/12

0.5833333333333334

RBF test success rate: 0/12

0.0

Poly test success rate: 1/12

0.08333333333333333

Sig test success rate: 11/12

0.9166666666666666

Phone 21: 80/20

Linear test success rate: 1/1

1.0

RBF test success rate: 0/1

0.0

Poly test success rate: 1/1

1.0

Sig test success rate: 0/1

0.0

Phone 21: 70/30

Linear test success rate: 1/2

0.5

RBF test success rate: 0/2

0.0

Poly test success rate: 0/2

0.0

Sig test success rate: 2/2

1.0

Phone 21: 30/70

Linear test success rate: 1/1

1.0

RBF test success rate: 0/1

0.0

Poly test success rate: 0/1

0.0

Sig test success rate: 0/1

0.0

Phone 22: 80/20

Linear test success rate: 3/4

0.75

RBF test success rate: 1/4

0.25

Poly test success rate: 3/4

0.75

Sig test success rate: 0/4

0.0

Phone 22: 70/30

Linear test success rate: 4/6

0.6666666666666666

RBF test success rate: 1/6

0.16666666666666666

Poly test success rate: 4/6

0.6666666666666666

Sig test success rate: 1/6

0.16666666666666666

Phone 22: 30/70

Linear test success rate: 7/14

0.5

RBF test success rate: 1/14

0.07142857142857142

Poly test success rate: 1/14

0.07142857142857142

Sig test success rate: 11/14

0.7857142857142857

Phone 23: 80/20

Linear test success rate: 2/5

0.4

RBF test success rate: 0/5

0.0

Poly test success rate: 1/5

0.2

Sig test success rate: 3/5

0.6

Phone 23: 70/30

Linear test success rate: 3/7

0.42857142857142855

RBF test success rate: 1/7

0.14285714285714285

Poly test success rate: 1/7

0.14285714285714285

Sig test success rate: 4/7

0.5714285714285714

Phone 23: 30/70

Linear test success rate: 5/16

0.3125

RBF test success rate: 1/16

0.0625

Poly test success rate: 5/16

0.3125

Sig test success rate: 8/16

0.5

Phone 24: 80/20

Linear test success rate: 2/2

1.0

RBF test success rate: 0/2

0.0

Poly test success rate: 0/2

0.0

Sig test success rate: 2/2

1.0

Phone 24: 70/30

Linear test success rate: 2/3

0.6666666666666666

RBF test success rate: 0/3

0.0

Poly test success rate: 0/3

0.0

Sig test success rate: 3/3

1.0

Phone 24: 30/70

Linear test success rate: 0/5

0.0

RBF test success rate: 0/5

0.0

Poly test success rate: 2/5

0.4

Sig test success rate: 2/5

0.4

Phone 25: 80/20

Linear test success rate: 0/1

0.0

RBF test success rate: 0/1

0.0

Poly test success rate: 1/1

1.0

Sig test success rate: 0/1

0.0

Phone 25: 70/30

Linear test success rate: 0/2

0.0

RBF test success rate: 0/2

0.0

Poly test success rate: 1/2

0.5

Sig test success rate: 1/2

0.5

Phone 25: 30/70

Linear test success rate: 0/1

0.0

RBF test success rate: 0/1

0.0

Poly test success rate: 1/1

1.0

Sig test success rate: 0/1

0.0

Phone 26: 80/20

Linear test success rate: 3/6

0.5

RBF test success rate: 2/6

0.3333333333333333

Poly test success rate: 1/6

0.16666666666666666

Sig test success rate: 3/6

0.5

Phone 26: 70/30

Linear test success rate: 4/9

0.4444444444444444

RBF test success rate: 2/9

0.2222222222222222

Poly test success rate: 1/9

0.1111111111111111

Sig test success rate: 6/9

0.6666666666666666

Phone 26: 30/70

Linear test success rate: 12/20

0.6

RBF test success rate: 3/20

0.15

Poly test success rate: 2/20

0.1

Sig test success rate: 15/20

0.75

Phone 27: 80/20

Linear test success rate: 1/4

0.25

RBF test success rate: 0/4

0.0

Poly test success rate: 0/4

0.0

Sig test success rate: 4/4

1.0

Attack Session Results:

Linear SVM predictions: [-1 -1 -1 -1 1]

Linear EXP success rate: 1/5

0.2

RBF SVM predictions: [-1 -1 1 -1 1]

RBF EXP success rate: 2/5

0.4

Poly SVM predictions: [-1 -1 -1 1 -1]

Poly EXP success rate: 1/5

0.2

Sig SVM predictions: [-1 1 1 1 1]

Sig EXP success rate: 4/5

0.8

Phone 27: 70/30

Linear test success rate: 3/6

0.5

RBF test success rate: 1/6

0.16666666666666666

Poly test success rate: 0/6

0.0

Sig test success rate: 5/6

0.8333333333333334

Attack Session Results:

Linear SVM predictions: [1 -1 1 1 1]

Linear EXP success rate: 4/5

0.8

RBF SVM predictions: [-1 -1 1 -1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [1 -1 -1 1 -1]

Poly EXP success rate: 2/5

0.4

Sig SVM predictions: [-1 1 1 1 1]

Sig EXP success rate: 4/5

0.8

Phone 27: 30/70

Linear test success rate: 7/13

0.5384615384615384

RBF test success rate: 0/13

0.0

Poly test success rate: 3/13

0.23076923076923078

Sig test success rate: 10/13

0.7692307692307693

Attack Session Results:

Linear SVM predictions: [1 -1 -1 -1 -1]

Linear EXP success rate: 1/5

0.2

RBF SVM predictions: [-1 -1 1 -1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [1 -1 -1 -1 -1]

Poly EXP success rate: 1/5

0.2

Sig SVM predictions: [-1 1 1 1 1]

Sig EXP success rate: 4/5

0.8

Phone 28: 80/20

Linear test success rate: 2/6

0.3333333333333333

RBF test success rate: 3/6

0.5

Poly test success rate: 1/6

0.16666666666666666

Sig test success rate: 5/6

0.8333333333333334

Phone 28: 70/30

Linear test success rate: 0/8

0.0

RBF test success rate: 4/8

0.5

Poly test success rate: 2/8

0.25

Sig test success rate: 6/8

0.75

Phone 28: 30/70

Linear test success rate: 11/19

0.5789473684210527

RBF test success rate: 0/19

0.0

Poly test success rate: 3/19

0.15789473684210525

Sig test success rate: 16/19

0.8421052631578947

Phone 29: 80/20

Linear test success rate: 4/5

0.8

RBF test success rate: 2/5

0.4

Poly test success rate: 1/5

0.2

Sig test success rate: 4/5

0.8

Phone 29: 70/30

Linear test success rate: 1/7

0.14285714285714285

RBF test success rate: 4/7

0.5714285714285714

Poly test success rate: 0/7

0.0

Sig test success rate: 6/7

0.8571428571428571

Phone 29: 30/70

Linear test success rate: 1/16

0.0625

RBF test success rate: 1/16

0.0625

Poly test success rate: 5/16

0.3125

Sig test success rate: 10/16

0.625

Phone 30: 80/20

Linear test success rate: 1/4

0.25

RBF test success rate: 2/4

0.5

Poly test success rate: 0/4

0.0

Sig test success rate: 3/4

0.75

Phone 30: 70/30

Linear test success rate: 1/6

0.16666666666666666

RBF test success rate: 3/6

0.5

Poly test success rate: 0/6

0.0

Sig test success rate: 4/6

0.6666666666666666

Phone 30: 30/70

Linear test success rate: 5/14

0.35714285714285715

RBF test success rate: 0/14

0.0

Poly test success rate: 4/14

0.2857142857142857

Sig test success rate: 9/14

0.6428571428571429

Phone 31: 80/20

Linear test success rate: 3/9

0.3333333333333333

RBF test success rate: 4/9

0.4444444444444444

Poly test success rate: 2/9

0.2222222222222222

Sig test success rate: 7/9

0.7777777777777778

Phone 31: 70/30

Linear test success rate: 8/13

0.6153846153846154

RBF test success rate: 5/13

0.38461538461538464

Poly test success rate: 1/13

0.07692307692307693

Sig test success rate: 12/13

0.9230769230769231

Phone 31: 30/70

Linear test success rate: 16/29

0.5517241379310345

RBF test success rate: 5/29

0.1724137931034483

Poly test success rate: 5/29

0.1724137931034483

Sig test success rate: 17/29

0.5862068965517241

Phone 32: 80/20

Linear test success rate: 4/8

0.5

RBF test success rate: 3/8

0.375

Poly test success rate: 1/8

0.125

Sig test success rate: 7/8

0.875

Attack Sessions Results:

Linear SVM predictions: [-1 -1 1 -1 1]

Linear EXP success rate: 2/5

0.4

RBF SVM predictions: [-1 -1 -1 1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [-1 -1 -1 -1 -1]

Poly EXP success rate: 0/5

0.0

Sig SVM predictions: [1 1 1 1 1]

Sig EXP success rate: 5/5

1.0

Phone 32: 70/30

Linear test success rate: 5/12

0.4166666666666667

RBF test success rate: 3/12

0.25

Poly test success rate: 2/12

0.16666666666666666

Sig test success rate: 10/12

0.8333333333333334

Attack Session Results:

Linear SVM predictions: [-1 1 -1 -1 -1]

Linear EXP success rate: 1/5

0.2

RBF SVM predictions: [-1 -1 -1 1 -1]

RBF EXP success rate: 1/5

0.2

Poly SVM predictions: [-1 -1 -1 -1 -1]

Poly EXP success rate: 0/5

0.0

Sig SVM predictions: [1 1 1 1 1]

Sig EXP success rate: 5/5

1.0

Phone 32: 30/70

Linear test success rate: 15/28

0.5357142857142857

RBF test success rate: 3/28

0.10714285714285714

Poly test success rate: 6/28

0.21428571428571427

Sig test success rate: 14/28

0.5

Attack Session Results:

Linear SVM predictions: [1 1 -1 1 1]

Linear EXP success rate: 4/5

0.8

RBF SVM predictions: [-1 -1 -1 -1 -1]

RBF EXP success rate: 0/5

0.0

Poly SVM predictions: [1 -1 -1 1 -1]

Poly EXP success rate: 2/5

0.4

Sig SVM predictions: [1 1 1 1 1]

Sig EXP success rate: 5/5

1.0

Phone 33: 80/20

Linear test success rate: 1/2

0.5

RBF test success rate: 0/2

0.0

Poly test success rate: 0/2

0.0

Sig test success rate: 1/2

0.5

Phone 33: 70/30

Linear test success rate: 1/2

0.5

RBF test success rate: 0/2

0.0

Poly test success rate: 0/2

0.0

Sig test success rate: 1/2

0.5

Phone 33: 30/70

Linear test success rate: 0/5

0.0

RBF test success rate: 0/5

0.0

Poly test success rate: 0/5

0.0

Sig test success rate: 0/5

0.0

Phone 34: 80/20

Linear test success rate: 0/3

0.0

RBF test success rate: 1/3

0.3333333333333333

Poly test success rate: 0/3

0.0

Sig test success rate: 2/3

0.6666666666666666

Phone 34: 70/30

Linear test success rate: 3/5

0.6

RBF test success rate: 0/5

0.0

Poly test success rate: 1/5

0.2

Sig test success rate: 1/5

0.2

Phone 34: 30/70

Linear test success rate: 2/10

0.2

RBF test success rate: 0/10

0.0

Poly test success rate: 1/10

0.1

Sig test success rate: 9/10

0.9

Phone 35: 80/20

Linear test success rate: 1/2

0.5

RBF test success rate: 0/2

0.0

Poly test success rate: 0/2

0.0

Sig test success rate: 2/2

1.0

Phone 35: 70/30

Linear test success rate: 1/3

0.3333333333333333

RBF test success rate: 0/3

0.0

Poly test success rate: 0/3

0.0

Sig test success rate: 3/3

1.0

Phone 35: 30/70

Linear test success rate: 0/6

0.0

RBF test success rate: 0/6

0.0

Poly test success rate: 3/6

0.5

Sig test success rate: 3/6

0.5

Phone 36: 80/20

Linear test success rate: 0/2

0.0

RBF test success rate: 1/2

0.5

Poly test success rate: 0/2

0.0

Sig test success rate: 1/2

0.5

Phone 36: 70/30

Linear test success rate: 1/3

0.3333333333333333

RBF test success rate: 0/3

0.0

Poly test success rate: 1/3

0.3333333333333333

Sig test success rate: 3/3

1.0

Phone 36: 30/70

Linear test success rate: 5/7

0.7142857142857143

RBF test success rate: 0/7

0.0

Poly test success rate: 3/7

0.42857142857142855

Sig test success rate: 6/7

0.8571428571428571