Supervised Learning Techniques in Mobile Device Apps for Androids

Priyanka Basavaraju and Aparna S. Varde
Department of Computer Science, Montclair State University, NJ, USA
basavarajup1@montclair.edu, vardea@montclair.edu

ABSTRACT

Mobile devices have become an integral part of our daily lives. Most people carry smartphones today almost everywhere; and have other mobile devices such as tablets, often more convenient than full-fledged laptops for work transit, short trips etc. This had led to development of apps for mobile devices, easy to download and access anywhere anytime. An important field improving human experiences on mobile devices is machine learning. This constitutes technqiues involving acquisition of knowledge, skills and understanding by machines from examples, guidance, experience or reflection to learn analogous to humans. Among learning paradigms herein, supervised learning comprises situations where labeled training samples are provided to administer the process, making it more regulated, similar to human instructors providing such examples with notions of correctness to guide human learners. Supervised learning techniques are useful in designing mobile apps as they entail guided examples capturing specific human needs and their reasoning in activities, e.g., classification. This paper gives a comprehensive review of a few useful supervised learning approaches along with their implementation in mobile apps, focusing on Androids as they constitute over 50% of the global smartphone market. It includes description of the approaches and portrays interesting Android apps deploying them, addressing classification and regression problems. We discuss the contributions and critiques of the apps and also present open issues with the potential for further research in related areas. This paper is expected to be useful to students, researchers and developers in mobile computing, human computer interaction, data mining and machine learning.

1. INTRODUCTION

Many problems can easily be solved by computers, from simple computations that involve multiplying billions of numbers in few minutes to the complex computations of calculating the distance of asteroids in the galaxies. These computations seem quick and easy to computers when compared to the speed and processing power of humans. On the other hand, a simple task for a human is to recognize or differentiate between different classes of objects such as trees, cars, buildings and people; even though these may not always remain the same throughout. For example, a broken chair with three legs would easily be identified by a human as a chair, a STOP sign covered in snow is still clearly distinguished as the given sign, a dog wearing a floral headgear would be recognized as a dog etc. See Figure 1 for a few examples of objects in the real-world that are easily identifiable by human beings (Image Source: Google). This identification process is so intuitive that it seems almost effortless to humans. However, for computers, such tasks seem really complex to execute since things tend to change in the real world [20]. Consider the example of a broken chair. A computer might fail to recognize this as a chair unless it is programmed with the worst case scenarios (otherwise, if a piece of furniture has three legs, it is more likely to be identified as a stool than a chair). Hence, instead of writing specific programs to solve such invidiual problems, scientists try to make computers solve these catergories of problems using examples provided through certain techniques. Such an approach is a part of the field of machine learning, where machines can learn analogous to humans in guided and / or intuitive ways [1]. An effective comprehension of how to make machines, more specifically, computers learn has led to numerous new users of computers with new levels of competence and customization [10]. Furthermore, a detailed comprehension of data processing algorithms for machine learning has prompted a superior understanding of human learning capacities as well. Much research has been conducted to make computers learn nearly as well as humans. Algorithms have been created that are successful for certain types of learning tasks. For various problems ranging from text recognition, speech recognition and object recognition [20], it has been found that machine learning algorithms typically perform better than other prior computational approaches.

Figure 1. Example of objects as represented in real world

Accordingly, machine learning techniques have proved to be effective for mobile devices like smartphones, sensors, handheld and automotive computing systems. Smartphones are equipped with mobile sensors that are widely available, e.g., cameras, gyroscope, GPS and accelerometers that are becoming cheaper and widely available [11]. Thus, mobile application developers claim that a smartphone constitutes a *"sensor"* carried by humans that can be further exploited to provide even more services to users, e.g. safety features [12].

In order to validate the lateral growth of associated research publications, we searched publications containing the phrases "machine learning" and "mobile applications" in the digital libraries of ACM and IEEE, by filtering the results based on the year published, ranging from 2006 to 2016. The results including machine learning implementation in mobile devices and mobile applications can be seen in Figure 2. It can be seen that there is a peak in 2008, possibly due a surge in the use of mobile devices by then [9]. The publication trend continues to be high, depicting the continued interest in research and development in the area of machine learning for mobile apps.

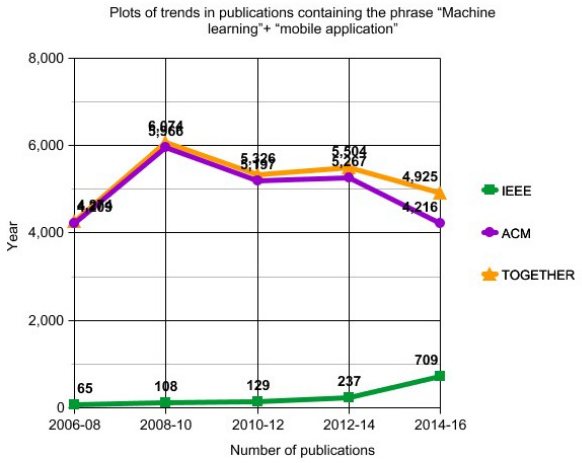


Figure 2. Publication trends in the areas of machine learning and mobile applications

It is thus useful to present a survey on machine learning techniques for mobile apps, which provides the motivation for our paper. In this survey paper, we fathom the techniques behind the tremendous growth in the machine learning field for decades, with particular emphasis on its use in recent technologies that seem almost indispensable, namely, mobile device apps. In order to make the survey more specific, we focus on supervised learning. This encompasses guided learning wherein labeled training data along with notions of correctness are provided in advance for the machine to learn. It is analogous to being taught by human teachers who are very specific in providing guided examples for learning with clear ideas of the right and the wrong that helps students learn in a regulated manner (as opposed to making them learn intuitively through trial and error, i.e., unsupervised learning, also a machine learning paradigm). It has been found that supervised learning techniques [15] are the more commonly used ones in mobile apps. This can be justified as follows. Mobile apps are designed for convenient, quick and easy access by humans. The guided learning processes of humans therefore seem to be more useful in incorporating their judgemeant and reasoning in the design of the apps. For making this survey paper even more focused, we consider the Android family of apps. This is because, based on the statistics, we observe that smartphones occupying around 52.8% of the worldwide market share today are Android devices [19]. Hence, to the best of our knowledge, a review of Android apps would be somewhat more useful. Thus, this survey paper concentrates on supervised learning techniques and their implementation in building Android mobile device apps. However, many of the details presented here with respect to techniques and applications can be helpful to other app developers and related professionals as well.

The rest of the paper is organized as follows. Section 2 provides an overview of supervised learning with selected approaches, namely, Support Vector Machines, AdaBoost, Artificial Neural Networks, Gaussian Process, Decision Trees and Naïve Bayes, all useful in mobile apps. Section 3 gives details of six Andorid apps in the six different areas of pedestrian help, fruit ripeness, application prediction, malware scanning, house pricing and healthcare monitoring that would be beneficial to Android users. It includes the data description, app learning details as well as contributions and critiques. Section 4 provides a discussion on the apps considering their utility value and open issues for further research. Section 5 states the conclusions.

2. SUPERVISED LEARNING METHODS

Machine learning paradigms fall in the categories of supervised learning, unsupervised learning and reinforcement learning as shown in Figure 3 herewith [10]. Among these, the supervised learning paradigm, our focus in this survey, consists of algorithms that learn a model from externally supplied instances of known data and known responses to result in a general hypothesis, such that the learned model can be used over new data to predict responses about future instances [15]. This is illustrated in Figure 4 (Image Source: MATLAB). In other words, such learning is performed with examples. The system is given existing data with examples that have been assigned one or more labels based on their input values and response values. These examples are used to train the system for the function it is expected to learn. The machine learns the given function and is then able to work on new data [15]. A simple analogy is a child identifying new fruits after being trained using existing fruits with their correct classification given in advance, e.g., fruit *x* is an apple, fruit *y* is an orange etc.

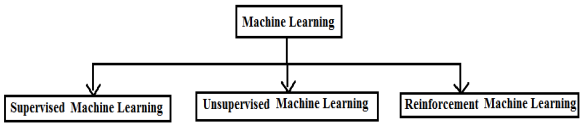


Figure 3. Machine learning paradigms

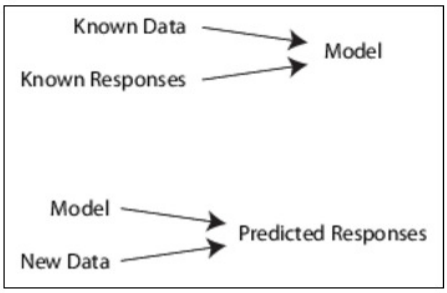


Figure 4. Graphical representation of supervised learning

Supervised learning includes two categories of algorithms:

1. Classification: This is used when the response values have differentiate between various discrete classes.
2. Regression: This is used when the response values are continuous, e.g., numerical data.

Some of the techniques in the supervised learning paradigm include: Support Vector Machines (SVM), Naïve Bayes Classifier, k - Nearest Neighbors (kNN), Decision Trees, Linear Regression and Nonlinear Regression, to name a few [15]. We focus on describing selected learning techniques that are useful in the mobile apps we discuss in this paper.

2.1 Support Vector Machines (SVM)

The technique of Support Vector Machines or SVM is a supervised learning method for classification proposed by Vladimir Vapnik and colleagues [6]. The goal of the SVM is to build decision planes or hyper planes that classify all training vectors into two different classes. The data for the training consists of a set of points that form the vectors. The learning process includes first training a support vector machine and then cross validating the classifier, explained as follows. Suppose we have two features x1 and x2 and have to classify the given input into either of the classes. The SVM draws hyperplanes that can separate the two different classes. The best classification is made by choosing the hyperplanes that leave maximum margins from all the classes.

Figure 5 provides a depiction of this. The SVMs make use of a nonlinear mapping function (Φ) that transforms the input data space to data in a feature space in such a manner as to make a problem linearly separable. SVM learning is used for data analysis and pattern identification, useful for classification and regression analysis [6]. Some practical applications of SVM are gene expression, text classification, image identification etc. SVMs are considered to provide good generalization accuracy and are fast in learning.

However, training a support vector machine causes a quadratic optimization problem with bound constraints and a linear equality constraint. Given this, there are many other issues to consider in designing the SVM learner. In particular, when training the machine with large sets of examples, the optimization techniques for general quadratic programs quickly become intractable in memory and time requirements.

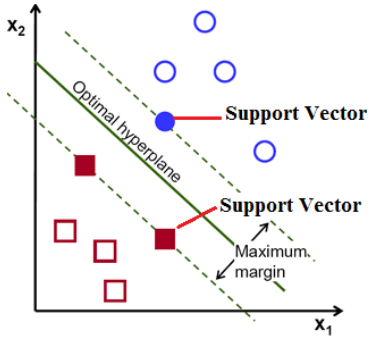


Figure 5. SVM showing the hyperplane

2.2 AdaBoost

The AdaBoost process is short for Adaptive Boosting and is a classification technique proposed by Yoav Freund and Robert Schapire [8]. Boosting is an ensemble method of creating a sequence of complex predictors made from blocks that are simple predictors. AdaBoost makes the model learn as follows. It sequentially trains a new model based on the errors of the previous model. In general terms, it builds a predictor model using any prediction algorithm and find the errors of the model results; it then focuses on these errors while building the next prediction model. This procedure helps in identifying the data points that are hard to predict and helps the latter classifiers in getting these examples correct. In the end, all the classifier models are combined together using a weighted combination. Models with errors are called weak classifiers. This learning process is represented in Figure 6.

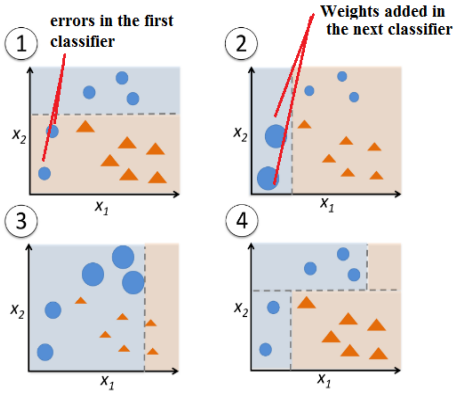


Figure 6. Graphical representation of AdaBoost

AdaBoost is often used for the following reasons [8]

1. Speed: It is fast, simple and easy to program.
2. Flexibility: It can be combined with any learning algorithm to form weak classifiers.
3. Effective: It produces the desired impacts, provided can consistently find rough rules of thumb.
4. Versatile: It can be used with data that is textual, numeric, discrete etc.

The classifier model with AdaBoost is easy to build; thus it is are highly used in applications with very large data sets. It requires no complicated iterative parameter estimation.

2.3 Artificial Neural Network (ANN)

Artificial neural networks serve as a model of the real neural network that exists in the human brain. They provide a general, practical method for learning real-valued, discrete-valued, and vector-valued functions from given training examples. The model is influenced by the observation of a human brain system that consists of very complex webs of interconnected neurons. Artificial Neural Network models comprise nonlinear computational components or processing units that work in parallel and are arranged in patterns resembling biological neural networks [4]. Nodes that form the computational components are connected by links that have weights. Once the model has been trained and tested, the predicted output should

ideally correspond to the actual output from the approximate mapping. An example of ANN appears in Figure 7.

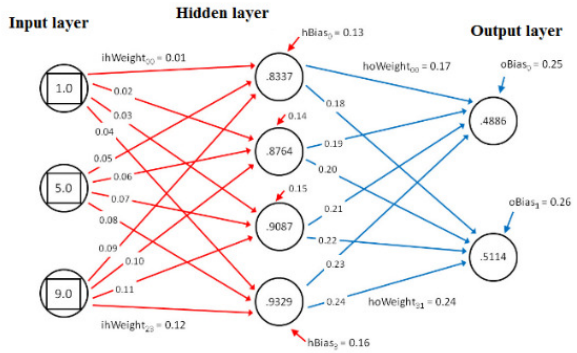


Figure 7. Example of an Artficial Neural Network

The ANN approach is known for its robustness to errors in the training data. A few of the applications that implement ANN recognize handwritten characters, human speech and robot control strategies. A very well-known ANN algorithm is the classical backpropagation method which uses gradient descent to tune network parameters in order to best fit a training set of input-output pairs [10, 4].

2.4 Gaussian Process

The Gaussian Process forms a simple regression model. A Gaussian Process (GP) can be defined as a collection of random variables, such that any finite number of them have a joint Gaussian distribution specified by its mean and covariance functions, i.e., vector and matrix, respectively [1]. Note that a Gaussian distribution also known as a normal distribution is a common continuous probability distribution often used to depict real-valued random variables whose distributions are unknown. It is a continuous function that approximates the exact binomial distribution of events. Training the GP model involves a model selection or a discrete choice between different functional forms for mean and covariance functions, followed by the adaptation of the hyper parameters of these functions. Hence, problems such as predicting prices, weather forecasting, disease tracking can deploy regression, since they typically involve estimating values that fall within a continuous range of outputs [1]. Figure 8 shows an example of a Gaussian model. In this example, the model is used for word prediction among a continuous range of possible words.

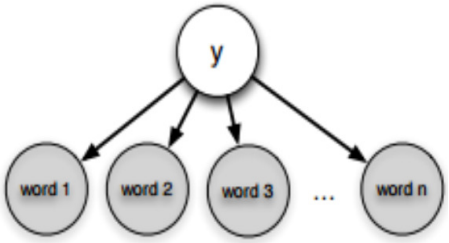


Figure 8. Example of a Gaussian model

2.5 Decision Tree

A Decision Tree learner is one of the most widely used practical methods for inductive inference [17]. This efficient nonparametric method, can be used for both classification and regression. It has a hierarchical data structure implementing a divide-and-conquer strategy for supervised learning where the local region is identified by a sequence of recursive splits in a smaller number of steps. A decision tree is composed of internal decision nodes, i.e., it classifies instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Figure 9 portrays a typical decision tree example used in software tools and text books [17]. Here various aspects of the weather are used to predict whether a game of tennis should be played.

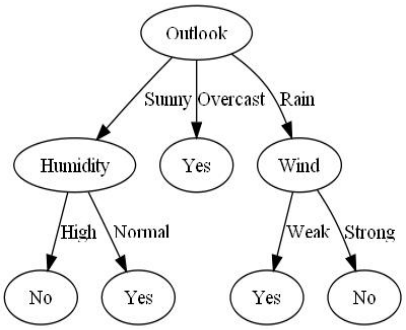
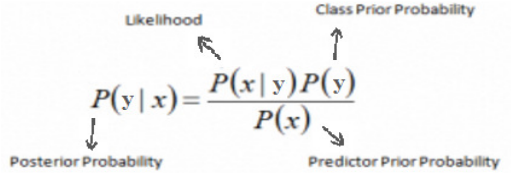


Figure 9. Decision Tree example for tennis playing

There are many problems that can be addressed using the decision trees. Here are some characteristics of problems that typically incorporate decision tree classifiers. 1. Instances are represented by attribute-value pairs; 2. The target function has discrete output values; 3. Disjunctive descriptions may be Required; 4. The training data may contain errors; and 5. The training data may contain missing attribute values [10].

2.6 Naïve Bayes

The Naïve Bayesian classifier is based on Bayes theorem proposed by Thomas Bayes (1702 - 1761); hence it can be stated that this is a conventional paradigm that has been around since the late 18th century. Bayesian classifiera are a part of the probabilistic family of classifiers [10]. They predict the class of a sample based on probability, i.e., they first predict the probability of the sample itself and then pick the class that has the highest probability given the observation. This is shown in the equation next.



The given equation provides a way of calculating posterior probability P(y|x) from P(y), P(x) and P(x|y). In practice, the denominator is often ignored since it does not affect the classification; the values of the features are given, so that the denominator is effectively constant.

Naive Bayes is a generative model that computes its predictions by modeling each class. Thus, given datasets with many attributes, it would be computationally expensive to compute P(x|y). In order to reduce the computation in evaluating P(x|y) P(y), the classifier assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This is called the class conditional independence assumption. It is made to simplify the computation involved and in this sense, it is considered *"naïve"*. The individual probabilities P(x1|y), P(x2|y), . . ., P(xn|y) can be estimated from the given training set.

Naïve Bayes classifiers are often used for the following reasons:

1. The model is simple, fast, space efficient and is convenient to build.
2. It handles real, discrete and streaming data well.
3. It is particularly used for dealing with large datasets as it does not need iterative parameter estimation.
4. The model is not sensitive to irrelevant features.
5. This classifier performs surprisingly well and outperforms other classification methods in many complex real-world situations.

The Naïve Bayes classifier serves as a popular method for text categorization, spam filtering and sentiment analysis [10]. Other applications include building recommendation systems to filter unseen data and real-time multiclass prediction systems such as face recognition.

Having thus described several pertinent classifiers, we now explain their usefulness in mobile app design with interesting real-world examples from the Android app family.

3. IMPLEMENTATION IN MOBILE APPS

This section describes a the implementation of some machine learning techniques for six mobile apps in the Android family of devices. In the first three apps shown here in the areas of pedestrian help, fruit classification and application prediction respectively, the model is trained by being embedded in the mobile devices. For the remaining three apps, in the areas of malware scanning, house pricing and healthcare monitoring respectively, the mobile devices are used for visualizing the data whereas the model training and data processing occur at the backend on local application servers.

3.1 Pedestrian Help App

According to recent data from the nonprofit National Safety Council, traffic fatalities raise closely up to 14% every year. Deaths and injuries in road traffic accidents pose a serious threat to global health and have a negative impact on social and economic progress [13]. Considering the death rate of traffic accidents involving pedestrians in 2009, a group of researchers have developed an Android mobile app for pedestrian help. This targets mobile users who get actively engaged with their mobile devices while walking on the streets. The developed app called *WalkSafe* is aimed to enhance pedestrian safety, by helping people that walk and talk, through alerts with a vibrate feature [21]. The app gives out a vibration signal once it detects an approaching vehicle, thus giving pedestrians the indication to be cautious. Since it has to predict accurate results in the real time, the app implements machine learning algorithms within itself to train the model. This application is a joint work of students, faculties and researchers from Computer Science Dartmouth College, USA and University of Bologna, Italy [21].

The contributions of their work are as follows. 1. The app includes intricate design with vehicle detection and pedestrian alert. 2. It incorporates machine learning algorithms on the phone to detect the front views and back views of moving vehicles. 3. It exploits the phone API (Application Programming Interface) to save energy by running the vehicle detection algorithm only during active calls and using mobile sensors such as a back camera to detect vehicles thet could be approaching the user. Figure 10 illustrates the WalkSafe app along with a suitable image of an approaching vehicle.

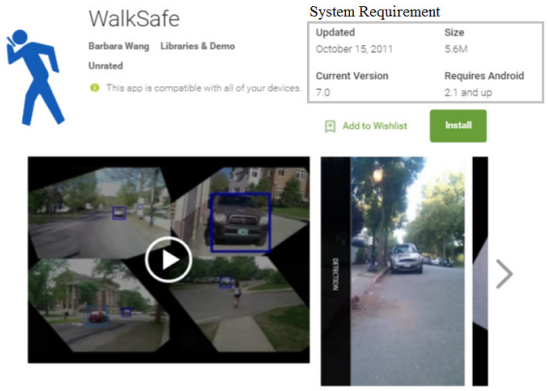


Figure 10. WalkSafe App with vehicle image

Their training dataset includes 7000 positive images (displaying the rear or front view of cars) and 3023 negative images (displaying pictures that show side views of cars or random urban environments).

As with many machine learning implementations, the design includes two stages, namely, the offline stage for training and the online stage for real time application, as described next.

Offline Stage: The core technology used in this project is the image recognition, that involves high computational intensive processing. Hence the application bases its vehicle recognition process on a model that is first trained offline; then uploaded and used for the online vehicle recognition. The process involves the following three steps. The first step consists of dataset building. In this step, images from different sources are collected to serve as the basis for further work by providing the training set. The second step involves training for image preprocessing. Most machine learning applications that involve images have to undergo preprocessing to extract the appropriate image with feature descriptors to train the classifier. Thus, 1032 positive samples are collected on which image processing techniques such as crop, resize, greyscale conversion and intensity variation are applied. This step creates 7000 different variations of the collected set, with the same sizes and alignments.

The third step entails feature extraction and classification. Here, a cascaded classifier that consists of several simpler classifiers is constructed based on Haar-like features. This is a well-known feature extraction technique often used because of its low computational property [21]. It works as follows. The process involves predefined sets of features, slid through the image from the top left to bottom right corner, reading pixel values. These features consider adjacent rectangular regions in a detection window; then sum up the pixel intensities in these regions and calculate the differences between them. The resulting value is called a Haar-like feature. The classifiers at every stage of the cascade pipeline "vote" on the classification of the same sample. These votes are then weighted according to the AdaBoost algorithm. The cascade classifier is trained on 20 stages. In addition to this, Decision Trees are used as weak classifiers to make the appropriate classification. The resulting Decision Tree is then uploaded and used for the online vehicle recognition.

Online Stage: In the online stage, similar steps are followed for the vehicle detection model that runs on the mobile device. In addition, we have the following functions for real-time processing in the app. The *image capture* function is for enhancing the phone performance and battery lifetime. The design team ensures triggering the image capture mode on the phone only during active phone calls. The application captures the information on its surroundings in the form of video frames. The *image preprocessing* function is available in real-time and its working is similar to that of the offline stage. The *feature extraction* function is for extracting important image features in real time. The relevant features are extracted and passed to the Decision Tree designed at the offline stage. If the prediction turns out to be positive, the application alerts the user by vibrating to inform the user about an approaching vehicle. Thus the online and offline stages together contribute to the effective performance of the WalkSafe app for Android mobile devices in the area of pedestrian help and safety.

The WalkSafe app is surely a useful innovation. However, we find that it could have certain drawbacks. For example, using the image processing technology could drain out the battery life of the mobile application completely if there are implementation problems. Also, observing the evaluation results provided in this work [21], the WalkSafe application detects cars upto 50 meters away from the pedestrians and provides just a few seconds for the pedestrian to react to the vibrate alert. The speed of the approaching car matters in such a situation, yet that aspect is not discussed in the paper [21]. Since the app involves camera, it will drain the phone camera early. This is reflected in the results shared by the team. The app has close to 10000 downloads and has received mixed reviews from the actual mobile phone users. Some of the users complain about no proper documentation on how the app works. All these issues provide the potential for further research, e.g., taking into account vehicle speed, providing user-friendly documentation and addressing the image processing issues such that they do not adversely affect battery life. However, on the whole, it has been found that WalkSafe is a good app for Android users.

3.2 Fruit Ripeness App

As mentioned in the introduction, smartphones come with sensors such as microphone, camera, GPS and accelerometer that enable them to interact with their environment in many ways. The application herewith exploits the nature of the microphone sensors. The role of the microphone is to collect audio signals and process them along with the necessary transformations when users are on the call. This nature of the microphone is used in an Android app called *Watermelon Classify* that is used to estimate the ripeness of watermelons [22]. During watermelon harvest, the actual ripeness of a given watermelon is identified by thumping on the watermelon to analyze the sound made by it. Note that once harvested, the watermelon will stop its ripening process and hence it becomes critically important to judge whether it is ripe or not prior to picking. A watermelon is around 90% water and the rest consists of sugar, an important indicator of the watermelon ripeness. The sugar content increases when watermelon becomes ripe and this leads to sound frequencies that differ in acoustic response from unripe watermelons. Knowing these factors, a group of researchers have developed an Android app that can differentiate between ripe and unripe watermelons. When a watermelon is patted, it produces an acoustic signal that can be captured by the mobile microphones. The signal undergoes various transformations to get its noise removed. Important features such as zero crossing rate, short-time energy, spectrogram, mel-frequency cepstral coefficients and brightness are extracted. These features are labeled and provided to a Support Vector Machine to train the model, for two different classes, ripe or unripe. The features extracted are known to accurately distinguish the watermelon classes based on earlier work of other researchers in the field. Once the model is trained, new input can be provided and the app can accurately classify the watermelon. Figure 11 shows the home screen of the Watermelon Classify app for Androids.

Figure 11. Layout of the Watermelon Classify app

The contributions of this work as seen in [22] are listed herewith. 1. The work involves understanding and extracting key acoustic features related to the ripeness of watermelons using relatively low-complexity measures including zero crossing rate, short-time energy and others. 2. The app implementing the machine learning technique of SVM for supervised learning is suitably harnessed here for effective results. 3. A crowdsourcing application has been designed by collecting user feedback to improve the classification methods. 4. The app achieves an accuracy of 89.9 % in correctly classifying ripe and unripe watermelons, which is very good.

The datasets in this application have been trained by thumping acoustic response signals. The relevant data has been collected

by patting on ten ripe and unripe watermelons, respectively. The features extracted for the training purposes are: zero crossing rate (ZCR), short-time energy (STE), sub-band short-time energy ratio, mel-frequency cepstral coefficients (MFCC), brightness and spectrogram [22].

While this is a very interesting app, it has some pitfalls, as we have noticed. It is a crowdsourcing application, in which the model has been trained with test data collected from 20 watermelons and has then been released to the users. The model which is embedded with the application files, receives the training data whenever a user runs the app, this process helps in refining the accuracy. With all the implementation, the app still provides 89% of accuracy, which is better than other other existing solutions. However, it may still not be enough to keep the harvest at stake for new farmers. Thus, further work with respect to crowdsourcing for enhancing the training data would help to make the app more accurate and generic.

3.3 Application Prediction App

This app targets users who have multiple applications installed on their smartphones and find it difficult to reach a particular application in the device, scrolling through the screens. The team in [3] has developed a *Home Screen App*. This home screen app acts as an intelligent layer between the underlying mobile operating system and the user interface that automatically organizes the installed apps in a more intelligent and personalized way; hence it also predicts the next app the user will deploy in near future.

The contributions of this work include the following. 1. For the implementation of the proposed idea, the team has conducted exhaustive studies on several domain-specific features for app usage prediction. 2. Considerably high accuracy is achieved using machine learning techniques. 3. The work proposes a new algorithm, i.e., a Parallel Tree Augmented Naïve Bayesian Network. This works around removing the strong assumptions of independence in Naïve Bayes for more accurate predictions.

In order to serve as the data for this app, two set of features are collected, i.e., human-engineered and automatic. For the human-engineered features, the data is collected by the Aviate sample log, where Aviate is a Yahoo based application. The log sample collected includes 60 million records with around 200,000 anonymized users, having a total of more than 70,000 unique apps. For the other type, namely, the automatic features, the sets of features collected include the real time spatiotemporal contexts as sensed by the home screen app.

Given this data, the next application is predicted incorporating two aspects. The first aspect is based on popularity, which assumes that the given user would be installing a concerned app for the first time and hence relies on its recognition among other users. The second aspect is based on the sessions, where various features are extracted based on the given user's spatiotemporal context collected using mobile sensors. These include Time, Latitude, Longitude, Speed, GPS Accuracy, Context Trigger Context Pulled, Charge Cable and Audio Cable. The app uses Parallel Tree Augmented Naive Bayesian Network (PTAN) proposed in this work as the prediction model. It builds upon Naïve Bayes and removes the strong assumptions of independence in Naïve Bayes, thus exploiting correlations among attributes. The training procedure involves two phases. Phase 1 involves structure training that learns the structure of the Bayesian network. Phase 2 comprises parallel parameters estimation that encompasses deducing a set of parameters for each individual user. Based on all these aspects, the next app that the user would install is predicted to assist the user in decision-making by taking into account overall popularity of apps as well as the individual user preferences.

This app thus constitutes research on how the home screen can be taken to the next level of predicting the app a given user might install in the near future. Note that the Aviate app is used here to collect log data and understand the attributes. Figure 12 shows an example of the home screen mobile app where 12(a) shows the changes through the day as automatically detected while 12(b) shows the auto-categorized apps that the user would be likely to install.

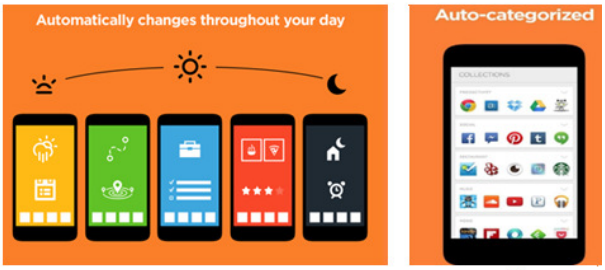


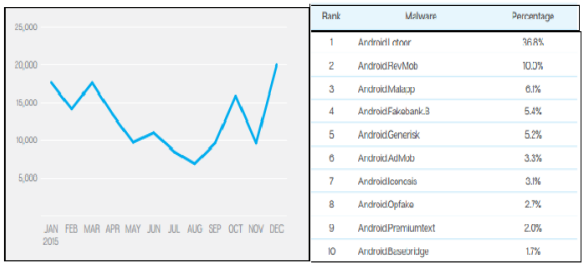
Figure 12. Example of home screen mobile app
(a) automatic changes in the day
(b) auto-categorization for app prediction

Like most other apps, this home screen app for application prediction also has its critiques. For this app, organizing the application based on user interests when the user installs the application for the first time is challenging. In this work, the team has already done enormous research in addressing the issues that occur due to the app or user cold start problem. The app cold start problem is that which occurs when a user installs a new app on their device. Likewise, the user cold start problem arises when a user installs and open the home screen app for the first time. Further research in this area could enhance the quality of the app even further.

3.4 Malware Scanning App

Malware applications tend to gain access to mobile devices with the intent of stealing data, damaging the device, annoying the user and other such issues. The attacker, i.e., the sender of malware manipulates the user into installing various harmful applications and / or increases unapproved remote access by exploiting the vulnerability of the device. Such applications give no lawful notification to the user. Their threat includes Trojans, worms, botnets, and viruses. Figure 13 depicts Android malware, such that 13(a) shows the projection of the overall malware volume while 13(b) shows a list of top ten malware examples. In order to address the malware problem, the design team in [5] claims that their proposed system can effectively perform real time malware detection on the Android platform. This application is a cloud based Android malware

analysis service called *ScanMe* mobile. The objective of this app is to bring about awareness of the Android application package (APK) files to the intended users before they make any attempt to install them on the given Android device. This is achieved by sharing a detailed report of the analyzed files to the user.



Rank	Malware	Percentage
1	Android1.otnor	36.8%
2	AndroidRevMob	10.0%
3	AndroidMalapp	6.1%
4	AndroidFakebank.3	5.4%
5	AndroidGenerisk	5.2%
6	AndroidAdMob	3.3%
7	AndroidIconosis	3.1%
8	AndroidOpfake	2.7%
9	AndroidPremiumtext	2.0%
10	AndroidBaseBridge	1.7%

(a) (b)

Figure 13. Android malware
(a) Projection of volume
(b) Top ten examples

There are important contributions of this work as stated next. 1. Though there exist other applications for malware detection, this work implements machine learning techniques that use training data to predict the malicious contents in the APK file, thus making the app more specific and useful. 2. ScanMe provides a plug and play solution that allows users to install their own local sandbox systems. These are application environments to analyze data locally by the user to facilitate faster and easier analysis of the APKs. 3. Apart from compiling a comprehensive report of the analyzed APK, the app shares such a report by publishing it through a Web interface that helps in bringing awareness of such malware applications to other users. 4. This app provides a static as well as dynamic analysis of the APK file which gives helpful information to the user. 5. The app incorporates advanced technologies and tools such as Google App Engine (GAE) datastore and its Website, Google Cloud Endpoint technology and Google Cloud Messaging (GCM) services to provide the best facilities to the users.

In order to provide data to train the model, the team has collected 100 known malware samples and 100 benign applications from the Android Malware Genome Project [2]. The training focuses on the Manifest.xml file which is the application file present in the APK. The team has a list of 206 Android permissions that are placed in three categories: *normal, dangerous and signature/system*. As the name suggests, the latter two categories can cause the most harm and have the most risky privileges.

The layout design of the ScanMe mobile app is portrayed in Figure 14. When users install the app for the first time, they are asked to register the device GCM service to establish a communication channel between the Android client application and GAE instances. If users choose to register the application, then it utilizes Google Cloud Endpoint technology to integrate mobile clients with a Google App Engine backend. This feature is implemented to conduct analysis of the APK file in the background and notify the users with messages when the analysis is completed. If the users fail to register, they have to login to the Website to obtain the results. Once the application is registered, the users can choose to upload one or more downloaded APK files (read from a Secure Digital memory card) into a sandbox server. This sandbox server is an Ubuntu system that analyzes the APK file. The system first runs a file, which extracts the permissions from the manifest file of the APK and sends it to the detection module.

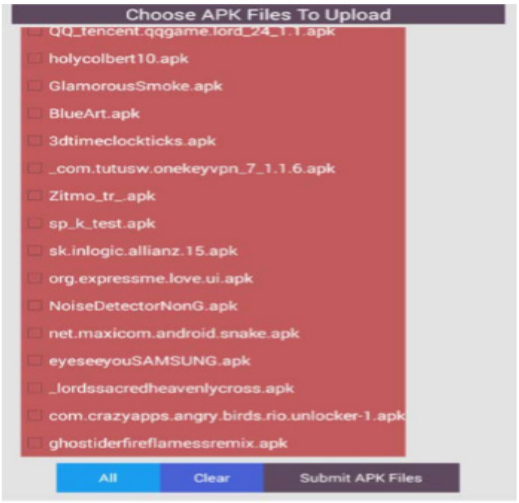


Figure 14. Layout design of ScanMe mobile app

The detection module is trained as follows. The sandbox identifies all the permissions and separates them into the three givevn categories of normal, dangerous and signature/system. It then lists all the custom permissions, features, services, receivers and activities, and writes all the data to the APK's report file. This data is fed to the Artificial Neural Network based malware detection classifier and is mapped to a format required by the ANN. The module learns patterns of permissions and gets trained to model a good classifier. The trained model is then used to carry out malware classification in the online detection process. Once the report is compiled at the sandbox, it is sent to the Google App Engine data store.

We have noticed that the paper in this work [5] has detailed information for implementing the approach, including clear architecture and methods. However, though the paper is well organized, it does not provide enough details on the features obtained to train the model, even though they seem critical. ScanMe Mobile users can locally scan APK files on the SD (Secure Digital) memory card of their phone. Yet, when the users download a mobile application, they are directed to the internal memory of the mobile device and there is a need to make necessary changes in the settings file forcing them to download APK files in the SD card. This app would have a lower utility value if users are unaware of this scenario. The app is also not found in the Google Play Database to verify the scan results. All these issues present the potential for further work, to enhance the appeal the malware scanning application.

3.5 House Pricing App

The housing market has always been a topic of interest. The trends in housing markets in any country are very interesting to house owners or potential buyers. Moreover, the trend also

affects the economic situations of the respective countries. There are many Web applications available that provide the trends of the housing markets and these have inspired the development of mobile apps for the same. We review herewith an app designed specifically for London users [14], given that London is a city with a very high cost of living and yet is one of the most sought after cities for residential purposes, having a diverse international population. The objectives of this app are to: 1. Predict future price changes of London properties that fall within a user-defined search radius. 2. Refer locations to the user that can be considered within a search radius based on the budget provided by the user.

This app has multiple contributions 1. The app handles a large set of data and produces accurate results in terms of future price changes of London properties. This is accomplished by the implementation of machine learning techniques. 2. The prediction is often trained with updated values of the housing market; hence it provides the latest information. 3. Most data processing occurs at the server side of the application, thus decreasing the computational task on the client side, thereby not adversely affecting the mobile device performance.

In order to generate the data in the app, the design team has collected the details of all property transactions that have occurred in Greater London from 1995 to 2013, bringing up the total to more than 2.4 million in the period. The data collected has been classified into different feature values, ID (Transaction ID), Date (Date processed, Month of transaction, Year of transaction), Transaction Price, Property classification (Type, Build, Tenure etc.), Address information (Postcode, Local authority, Full address, Borough, Ward etc.).

Figure 15 shows the layout design of the house pricing app [14]. The app implementation involves two frameworks: client side and server side. The client side is developed to collect the information from the user, the area code, other search refinement information and budgets. All this data is then passed to the server side.

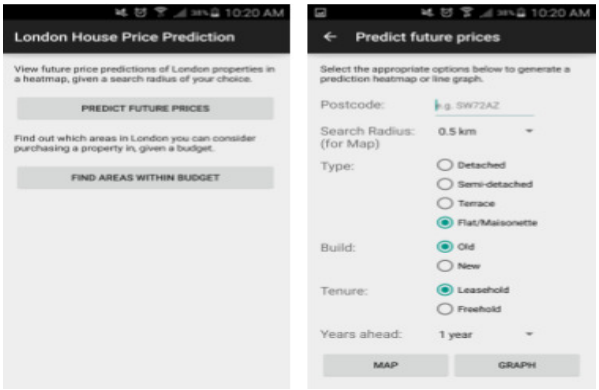


Figure 15. Layout design of house pricing app

The data gets processed at the server side and the results are displayed back to the client side in the form of map and graph information. The server side of this application contains two servers. The Web server acts as a middle layer and the application server is where the computations are handled.

The data training process involves the following [14]. In order to capture geographical variation, the map is represented in terms of latitudes and longitudes that are converted to appropriate postcode addresses. The values for the collected feature sets are noted. Since the data is too large, the designers use the technique of dividing the data into map grids and train the prediction model seperately. In the end, the predictions for each test input are made by combining the predictions of all the local models. This data is stored at the server end. The data is then sent to the application server where the model is trained and stored by Web server which acts as a middle layer to transfer the data.

It is noteworthy to observe the critiques of this app. The house pricing app deals with large sets of data, which is good for designing an effective classifier. However, the app may not be ideal for predicting the trends as the data is not updated on a regular basis. Note that the trends in the housing market can change within a day. According to their paper [14], it takes a day to train the model again with the new data since there are only a few servers. This flaw of a less number of servers has to addressed for catering better to the users. Also, the future price search options in the current app could be enhanced to provide more refinement choices for the users. These issues offer the scope for further work.

3.6 Healthcare Monitoring App

There are applications of machine learning techniques in the field of medicine. Sleeping disorder is an important problem; if not identified and treated right, it might lead to various health issues including chronic disorder, heart failures and in the worst case death [18]. According to the latest statistics from a leading hospital, around 70 million Americans suffer from disorders of sleep and wakefulness. Most of them happen to be undiagnosed. Researchers in [16] have contributed to this area by building a mobile app that can help people track sleeping patterns and take immediate action upon seeing disturbances. This app can also be used by doctors to remotely monitor their patients who may have sleep disorders.

This app entails significant contributions. 1. The work here involves developing a low-cost system for a robust, non-invasive multimodal sleep monitoring. 2. The app uses machine learning to automatically analyze the collected data, recognize sleep patterns and track the users' breathing behavior. 3. This particular app is also available for Apple iPhones in addition to Androids, which is an added advantage, increasing clientele.

Regarding data collection, this sleep pattern monitoring app is a continuation of prior work by the same researchers. They have designed a pressure mat capable of producing 1728 features based on the 1728 pressure sensors embedded inside the mattress. The values here can range from 0 to 100 for each feature. Thereafter, ten important data features can be selected such that they clearly identify different classes. The app also uses a Web camera for monitoring breathing behavior of the users. It observes chest movements by capturing frames of images for comparing the data with a given threshold value. Apart from these values, the team has 500 feature vectors collected by 5 different sleeping positions. This is done by

making various volunteers sleep on the mattress and thereby monitoring their sleeping positions, considering people of different sizes and heights. There are totally 2000 samples in the dataset. This app works in two stages, offstage and onstage as described next. Figure 16 portrays the layout design of this sleep pattern monitoring app [16].

Offstage: In this stage, the concerned data on sleep positions and related features is collected and the model is trained based on the input values and response values. Here, they use SVM, given that the app involves five different classes pertaining to sleep positions, namely, left, up, down, side and sitting. It is to be noted that SVM is usually used with two classes, however, when there are more than two classes, multiclass SVMs are used, as in this application. The breathing patterns are also monitored in the offstage.

Onstage: In this stage, the data is aggregated and then displayed to the people who are responsible for monitoring the sleep patterns. The app displays a graphical representation of the monitor information. This can be utilized by the concerned app users as well as by the doctors who would be interested in analyzing any disturbances.

Figure 16. Layout design of sleep monitoring app

Given that this app is used in healthcare, its critique becomes even more significant, since the data is highly sensitive. We have noticed the following negative aspects that deserve further attention. The app provides a graph that makes it quite hard for any new user to analyze the data. Moreover, the monitoring process involves setting up a huge lab which might make it very expensive to use. The application works well if the set-up is at the hospital and the users are given thorough knowledge on how to track their information to share it with the doctors. The issues listed herewith motivate further research in the area, in order to provide greater enhancement.

Having thus described various mobile apps with respect to their features, learning techniques and contributions, we now present a general discussion on these.

4. DISCUSSION

Mobile devices have become such an important part of our daily lives that they seem almost indispensable today. Accordingly, apps, providing quick and easy access on the fly to a variety of services, have become very popular on mobile devices such as smartphones. Among various mobile devices, the Android family is found to have the greatest market share in the world. Android OS is the most popular across the globe, considering the various mobile operating systems including iOS by Apple, Blackberry OS etc. Thus, mobile apps on Android devices deserve special attention. It is found that machine learning and data mining techniques, especially those in the category of supervised learners are very useful in designing mobile apps. Considering these factors, we have focused our survey on the use of supervised learning techniques in mobile device applications, with particular emphasis on Android apps.

Various apps are surveyed here and are found to be useful in several aspects of everyday life. We next discuss the utility value of each app from a user perspective.

The *pedestrian help app*, namely, *WalkSafe* is likely be very helpful to people in busy cities, e.g., New York City, London, Mumbai, Shanghai and so on. It is found that people are often on their smartphones, while waiting for buses, crossing streets, halting at signals etc. Thus, a safety signal given by WalkSafe would prove useful here. It could also be useful in smaller towns, however people are more likely to drive there than walk and also roads there are more clear.

The *fruit ripeness app* called *Watermelon Classify* seems a boon to farmers as it would assist them in detecting whether a given watermelon is ripe. Also, it could help shoppers who are not experts in estimating ripeness which could be useful as it would avoid buying low quality fruit or being cheated by shopkeepers. It would also serve as a simple learning tool for novices in the area of horticulture. The principles used here could potentially be extended to some other types of fruits as well.

The *application prediction app* could be helpful for almost all users, especially to those that tend to install several apps. This *Home Screen app* would help such users get organized by predicting the next app most likely to be useful to them based on overall popularity and individual user preferences. It would thus reduce excessive installation of apps, as many of them would seem low on the priority list. It would also save time by giving users suggestions rather than having them browse all available apps before making an installation decision.

The *malware scanning app* here aka the *ScanMe mobile app* appears to be almost a necessity on mobile devices. This could be considered analogous to having virus scanners and other malware detection tools on desktops and laptops that seem to be a must today. As mobile devices gain popularity, attackers find ways to cause trouble to good users. Thus, protective apps on handheld devices are very useful as they could prevent serious attacks and reduce the threat of harmful software.

The *house pricing app* appears to be a gift to those interested in seriously buying a house or window shopping for one. It could

also be useful to renters who could be potential home-owners. The *London housing app* described here would give users in that city an idea of what to expect in the housing market, preparing them for an informed discussion with a realtor, property owner or agent. Having such an app handy on their phone would help them get the information at-a-glance and help them negotiate better. On the other hand, the sellers could also benefit from this in addition to buyers by getting to know the market prices before offering a quote on their house. It would help them have more well-guided discussions with prospective clientele as well. While London has been selected here due to being an expensive yet popular metropolis for house-buying, the discussions applied here would be useful in potential housing apps for other towns and cities as well.

The *healthcare monitoring app* reviewed herewith is by far the most interesting and useful among the six apps discussed. Since it monitors *sleep patterns* of the users, it could help in early detection of sleep disorders, thereby preventing further health problems. This app is usable on Apple and Android devices, thus enhancing its utility value. Given that it is a medical app, the nature of the data is very sensitive, and this would be a risk-intensive app that needs to be used with caution, using doctor advice as needed, for greater effectiveness. While this app, like the others here, has it advantages, it also has certain concerns. These concerns present open issues for future work.

Accordingly, we now list some open issues pertaining to the various apps presented here that could potentially lead to further research. These are as follows.

- Incorporating vehicle speed while giving a vibrate signal to warn of a moving vehicle in the WalkSafe app
- Performing image processing in such a manner as to enhance battery life in this pedestrian help app
- Improving the crowdsourcing for the Watermelon Classify app to increase ripeness estimation accuracy
- Extending the app for ripeness of other fruits taking into account aspects of Watermelon classification
- Further addressing the app cold start problem in the Home Screen app (when the app is new to a given user), thus providing better service
- Fully solving the user cold start problem (when the user is new to a given app), in order to augment the usage of the application prediction app
- Conducting deeper studies on the verification of the malware scanning results in the ScanMe mobile app
- Making the download of this malware scanning app more user-friendly in order to enhance its utility
- Giving frequent data updates to the London housing app to incorporate recent trends, thus improving its own usefulness and also exemplifying the design for other house pricing apps
- Improving future price searches in the current app for offering more refinement choices to various users in this house pricing app and thus other similar apps
- Augmenting the interpretation of the results in the sleep pattern app, given that its present graph display is difficult to understand, especially for new users
- Heading towards making this healthcare monitoring app more cost-effective, since its current version seems expensive to implement, with lab costs etc.

Another point to note is that the healthcare app described here is generic to the Apple and Android families of mobile devices. It would be interesting to provide this facility in some other apps as well to make them more appealing to a wider range of users across multiple platforms. This calls for further research.

Hence, the various open issues listed herewith provide the potential for future work in the respective areas. They motivate further research and development that entails supervised learning techniques and mobile app design. This would involve studies by data mining and machine learning researchers in conjunction with those in fields such as human computer interaction (HCI) and mobile computing. Mobile device usage is ever-growing and thus inspires the development of more apps as well as the enhancement of existing apps, taking into account these and other related aspects.

5. CONCLUSIONS

In this survey paper, we present a brief overview of various supervised learning techniques useful in the implementation of mobile apps. We also provide examples of a few interesting mobile apps with specific reference to the Android family of handheld devices. We describe the design layout of the apps, and the respective learning technique(s) as deployed within the apps. We also outline a list of contributions of each app as entailed in the corresponding piece of research leading to its development. Furthermore, we offer a critique of each app from a user perspective. We also present a discussion on the apps overall, entailing their utility value and putting forth open issues that could potentially lead to further research. This survey paper would be useful to various professionals in areas such as human computer interaction (HCI) and mobile computing. It would also be useful to the machine learning and data mining communities, from the research as well as the development angle.

6. REFERENCES

[1] Alpaydin, Ethem, *Introduction to Machine Learning,* MIT Press, 2014.

[2] *Android Malware Genome Project*, URL reference: http://www.malgenomeproject. Org/

[3] Baeza-Yates, Ricardo., Jiang, Di., Silvestri, Fabrizio, and Harrison, Beverly, "Predicting The Next App That You Are Going to Use", *ACM International Conference on Web Search and Data Mining (WSDM),* 2015, New York, NY, USA, pp. 285-294.

[4] Bishop, Christopher. M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[5] Cole, Yevgeniy., Zhang, Hanlin., Ge, Linqiang., Wei, Sixiao., Yu, Wei., Lu, Chao., Chen, Genshe., Shen, Dan., Blasch, Erik, and Pham, Khanh D., "ScanMe mobile: a local and cloud hybrid service for analyzing APKs", *ACM Conference on Research in Adaptive and Convergent systems (RACS)*, 2015 New York, NY, USA, pp. 268-273.

[6] Cortes, Corinna and Vapnik, Vladimir. “Support Vector Networks”, *Machine Learning*, 1995, 20: 273-297.

[7] Felt, Adrienne Porter., Finifter, Matthew., Chin, Erika., Hanna, Steve, and Wagner, David, “A survey of mobile malware in the wild”, *ACM workshop on Security and Privacy in Smartphones and Mobile devices (SPSM)*, 2011 New York, NY, USA, pp. 3-14.

[8] Freund, Yoav and Schapire, Robert, "A short introduction to boosting", *Journal of the Japanese Society for Artificial Intelligence*, 1999, 14: 771-780.

[9] *History of Mobile Phones*, URL reference: http://www.knowyourmobile.com/nokia/nokia-3310/19848/history-mobile-phones-1973-2008-handsets-made-it-all-happen

[10] Michalski, Ryszard S., Carbonell, Jamie G. and Mitchell, Tom. M. *Machine learning: An Artificial Intelligence Approach*. Morgan Kaufmann, 1985.

[11] *Mobile App Survey*, URL reference: https://www.sans.org/reading-room/whitepapers/analyst/2013-mobile-application-survey-35080

[12] Nasar, J., Hecht, P., and Wener, R., “Mobile telephones, distracted attention, and pedestrian safety”, *Accident Analysis and Prevention*, 2008, 40(1), 69-75.

[13] National Safety Council, URL reference: http://www.nsc.org/pages/home.aspx?var=mnd

[14] Ng, Aaron, and Deisenroth, Marc, *Machine Learning for a London Housing Price Prediction Mobile Application*, Technical Report, June 2015, Imperial College, London, UK.

[15] Omary, Zanifa., Mtenzi, Fredrick, “Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning”, *International Journal for Infonomics*, 3:314–325, 2010.

[16] Papakostas, Michalis., Staud, James., Makedon, Fillia, and Metsis, Vangelis, “Monitoring breathing activity and sleep patterns using multimodal non-invasive technologies”, *ACM International Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, 2015, New York, NY, USA, Article 78, 4 pages.

[17] Quinlan, J.R., “Induction of Decision Trees”, *Machine Learning*, 1986, 1: 81-106.

[18] *SleepMed Publication*, URL reference: http://www.sleepmedsite.com/page/sb/sleep_disorders/sleep_statistics

[19] *Smartphone Market*, URL reference: https://www.comscore.com/Insights/Rankings/comScore-Reports-January-2016-US-Smartphone-Subscriber-Market-Share

[20] Viola, Paul, and Michael Jones, “Rapid object detection using a boosted cascade of simple features”, *Computer Vision and Pattern Recognition*, 2001, 1: 511–518.

[21] Wang, Tianyu., Cardone, Giuseppe., Corradi, Antonio., Torresani, Lorenzo, and Campbell, Andrew T., “WalkSafe: a pedestrian safety app for mobile phone users who walk and talk while crossing roads”, *ACM Workshop on Mobile Computing Systems & Applications (HotMobile)*, 2012, New York, NY, USA, Article 5, 6 pages.

[22] Zeng, Wei., Huang, Xianfeng., Arisona, Stefan Müller, and McLoughlin, Ian Vince, "Classifying watermelon ripeness by analysing acoustic signals using mobile devices". *Personal & Ubiquitous Computing*, 2014, 18(7): 1753-1762.