Abstract—Improving the accuracy, reducing the time to authenticate users and preserving privacy are some of the pivotal issues in smartphone security. A majority of published owner identification methods have concentrated on improving accuracy, emphasizing less on response time. Usage pattern of smartphone apps by the owner may be used as an important signature to differentiate between the legitimate user and others. In this work, we rank the most informative apps specific to an owner to identify the owner using an information theoretic approach. Interestingly, the reduced set of data based on highly ranked apps gives a higher detection accuracy with reduced learning time consumed by the classifiers.

I. INTRODUCTION

In the modern digital age, smartphones have become an integral part of our lives. Smart devices are no longer limited to being devices of communication for a few. Recent statistics show that in 2015, the number of active mobile phone subscriptions reached 7.216 billion worldwide, indicating the importance of these devices in our daily lives [1]. With the increasing popularity of these devices, the number of utility software programs, popularly called Apps (application), has also risen remarkably. A study reports that the number of Android apps jumped from 1 million to 2.4 million within the time span from 2013 to 2016 [2]. Although these devices and apps offer many significant uses, they also introduce many possible threats to owners of smart devices, related to security and privacy.

Much research has been conducted to improve security and privacy of owners’ confidential information stored in smartphone devices. Personal Identification Number (PIN) [3] or password based approaches are popular and common authentication methods. However, PIN based approaches are susceptible to theft due to improper use of the PIN by the owners. User behavior based active authentication [4] is an alternative to protect devices from misuse or information theft. Continuous authentication is a form of behavioral authentication used to identify the legitimate user of a device. Continuous authentication uses a behavioral biometric such as keystroke dynamics or touchscreen usage [4]. Although these methods provide promising results, they are also vulnerable to possible privacy theft. Several approaches use tracking of app names, phone numbers called and texted, and websites that users visit and the WiFi network connected to validate users. The use of such data elements may actually increase the possibility of breaching of user’s privacy.

Machine learning is an effective alternative for intelligent detection of fraudulent users of a smartphone. Continuous authentication based on a user’s usage pattern of a subset of apps may help prevent misuse of the device. Since the pattern can be tracked on the device itself, the privacy of the user will not be at stake. In this work, we design a machine learning framework considering informativeness and popularity of apps. We use an information theoretic measure to rank top apps and reduce the training set remarkably to train the classifier. A majority of machine learning approaches for legitimate owner identification do not consider the time involved to train the classifier for identification. Moreover, they do not take into account the frequencies of use of the apps used by legitimate users. We aim to keep instances of only the most informative apps in the training datasets to obtain better results in terms of EER and running time.

The rest of the paper is organized as follows. Section II outlines existing work on legitimate smartphone owner identification. Section III discusses on our proposed approach. Section IV reports experimental results. Section V summarizes our work with concluding remarks.

II. PRIOR RESEARCH

Smartphone users prefer to have a transparent authentication system in order to increase the level of security through continuous or periodical inspection [3]. Instead of PIN based security, user or owner usage behavior may be an effective alternative for continuous authentication. In continuous authentication, a user does not need to be explicitly activate security features or authenticate his/her legitimacy continuously or periodically. There is considerable research in developing effective behavioral authentication systems.

Shi et al. [5] use an implicit authentication policy by accumulating the user’s activities throughout the day to make a decision on a legitimate user. An authentication score is computed based on a user’s recent activities. If the score is below a certain threshold, the user is alternatively authenticated through a conventional PIN based method. However, monitoring every regular activity of the user is a breach of privacy and may also invite additional security threats.
A behavioral profiling based framework was introduced by Clarke et al. in [6], where various user inputs in terms of logs of personal SMS messages and calls, the locations of the user and mobile app activities are used to analyze the behavior of the smartphone owner. Based on historical usage, authors compute the feasibility of using app usage to validate users. The authors use the MIT dataset [7] with 20 subjects tracked for 26 days for app usage, telephone calls and SMS. The best obtained result for all users overall app usage is an EER of 9.8%. The authors implement and evaluate a proposed framework and obtain FRR 11.45% and FAR 4.17%.

Fridman et al. [8] collect a dataset with 200 participants for a period of 30 days with four modalities: texting, app usage, websites accessed and locations. The authors implement a binary classifier to decide whether a subject is legitimate or not. Characterizing and testing are based on five-fold cross-validation. The proposed method achieved EER of 5%.

Khan and Hengartner [9] introduce an application-centric implicit authentication approach, considering when and how apps are used. The authors recruit 32 subjects and consider the usage of four android apps: a browser, a map, the launcher and a comic viewer. The authors use Kullback-Leibler (KL) information divergence measure [10] to evaluate the proposed method, and obtain EER of 7.645%, 9.055%, 10.37% and 6.77% for launcher, browser, map and comic viewer, respectively.

Hayashi et al. [11] implement an approach using all-or-nothing classification considering use of sensitive apps. This approach asks 14 users about their preference among three different scenarios: always available (apps are available in both locked and unlocked devices), split (some functions for apps are available when the device is opened whereas the rest of the features are available when the device is closed) and after unlocked (apps are open when the device is locked). Each subject is asked to rank the 20 most important accessed apps, then to express their preference by classifying them into always available, split and after unlock. The authors found that subjects prefer 35% of their apps to be always available, 20% of apps to be split and 45% of apps to be available after unlocking the device.

Papamartzivanos et al. [12] introduce a client app called Crowdsource to improve the privacy for users who share their apps (SMS, Contacts, Photo, Location, Camera and Audio) for host and cloud sites. The approach is evaluated in terms of consumption of CPU and memory, and obtain noteworthy results.

Legitimate users of a smartphone are often prefer to use a subset of apps installed on their device. Analyzing app usage and discovering this subset of these important apps in terms of their frequent use may help identify the genuine owner of a cellphone. Machine learning approaches for detecting smartphone owners rely on publicly available datasets. Collecting such datasets is expensive, and since the number of users in these datasets is usually small, it may also lead to overfitting during classification. In this work, we try to rank the most informative or popular apps for instance reduction to improve the error rate with reduced training time.

III. RANKING APPS. USING MUTUAL INFORMATION

We propose a ranking approach to discover the most important apps based on their informativeness or prediction power in comparison to other apps. We measure the predictive power of the app using its mutual dependency with the class attribute. Non-informative apps do not contribute to improving prediction accuracy and hence one can remove training tuples related to these apps. The benefit of removal of such tuples are two-folds. It may overcome or minimize overfitting suffered by the classifiers. Second, it helps lower the learning time.

We use mutual information to measure the dependency between predictive variables and the class attribute. Mutual information (MI) in a particular environment is described as the amount of information that an event contains about the occurrence of another event [13]. It is a measure of the mutual dependence between two discrete (or continuous) random variables X and Y jointly distributed with probability \( p(x, y) \) and measures how similar the joint distribution is to the product of marginal distributions \( p(x)p(y) \). For a pair of discrete variables it can be calculated as:

\[
MI(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \tag{1}
\]

The Product of Mutual Information (PMI) of the dataset \( D = \{A_1, A_2, \ldots, A_{M-1}, C_i\} \) of \( (M-1) \) attributes, with \( C_i \) being the class attribute with \( p \) different class labels or user ID can be calculated as:

\[
\mathcal{PMI}(D) = \prod_{X=1 \cdots (M-1)} MI(X, C_i=p) \tag{2}
\]

where, \( X \) is the set of values for an attribute \( A_i \) in \( D \).

We measure the influence of the apps in terms of their predictability. The influence factor of an app is the indicator of how the PMI score of the original dataset \( \mathcal{PMI}(D) \) varies in absence of the tuples (or instances) related to the app.

**Definition 1:** (Influence Factor:) A dataset \( D = \{N \times M\} \) with \( N \) instances and \( (M-1) \) features is represented as vector \( D = < A_1, A_2, \ldots, A_{M-1}, C_i >. \) The \( M^{th} \) attribute is the class label. \( K \) is the number of instances or records in \( D \) containing the target app \( A \). The reduced dataset can be represented as \( D' \subseteq D \) with \( N' = N - K \) instances, i.e., \( D' \) contains all the records from \( D \) except \( K \) records related to \( A \).

The app or application \( A \) is influential if \( \mathcal{PMI}(D') > \mathcal{PMI}(D) \) holds.

We use a knockout approach to remove tuples related to each app one at a time and measure its influential factor. We rank an app \( A \) by comparing the PMI of the whole dataset with the reduced dataset by eliminating the instances related to \( A \).

**Definition 2:** (Predictability Rank:) Given \( \mathcal{PMI}(D) \) of the whole dataset \( D \) and \( \mathcal{PMI}(D') \) is the MI of the reduced
dataset \((D')\) with respect to \(\mathcal{A}\), the predictability rank of \(\mathcal{A}\) can be given as:

\[
\text{Rank}(\mathcal{A}) = PMI(D) - PMI(D').
\]  

When ranking any other app say, \(\beta\), the tuples related to app \(\mathcal{A}\) are added back to the original dataset \(D\). The first app in the ranking will be of the highest priority and most informative whereas apps with lower scores have little or no effect on predictability of owners or class in dataset \(D\). The most informative apps with higher ranks are used to reduce the original dataset. We keep only those instances that are related to higher rank apps. The rest of the instances are removed from the original dataset.

There is a possibility that during the process of data reduction based on rank of the apps, one may lose vital information. We address the issue by using a weighted ranking score and discuss the scheme next.

A. Weighted Ranking

Data reduction may lead to elimination of class information as well. This is because non-rank holder apps may be associated with some owners’ records. Elimination of the low priority app tuples may in turn eliminate completely the associated user ID (class variable). To obviate this possibility we introduce what we call the popularity index for an app. The popularity index is an indicator of the popularity of an app’s use by users. It helps in retaining a highly used app despite its mutual independence with the user or rank being low.

Definition 3: (Popularity Index) Given an app \(\mathcal{A}\), popularity index is the ratio of the number of owners who use the app \(\mathcal{A}\) with the total number of owners. The popularity Index of an app \((\mathcal{A})\) within the dataset \(D\) can be given as follows.

\[
\text{Popularity}(\mathcal{A}) = \frac{\text{Support}(\mathcal{A})}{\text{Number of Owners} \in D}.
\]

where, \(\text{Support}(\mathcal{A})\) is the number of owners who use the app \(\mathcal{A}\).

Based on the popularity index we calculate the weighted rank using following equation.

\[
w\text{Rank}(\mathcal{A}) = \text{Popularity}(\mathcal{A}) \times \text{Rank}(\mathcal{A}).
\]

Weighted rank is now able to take care of the popular apps with relatively lower PMI scores.

We normalize the rank of the apps using a rank normalization scheme as discussed below and finally rank the apps based on decreasing priority.

B. Rank Normalization

We normalize the rank scores in the rank list \(\mathcal{L}\) to the range \([0, 1]\) by normalizing the top ranked app as 1.

\[
\text{Norm}(\mathcal{A}_i) = \frac{1 - (\text{Rank}(\mathcal{A}_i) - \min(\mathcal{L}))}{\max(\mathcal{L}) - \min(\mathcal{L})},
\]

where, \(\max(\mathcal{L})\) and \(\min(\mathcal{L})\) are the maximum and minimum rank scores in the list \(\mathcal{L}\), respectively. \(\mathcal{L}\) is arranged in descending order based on normalized ranks for the entire dataset. We consider \(k\) top ranked apps and remove the rest by deleting all their instances in the original dataset to create a reduced dataset.

C. Removal of Apps with Low Information

Learning a classification model is more effective if a substantial number of examples or instances related to all classes are available for training. The process of ranking of apps, followed by removal of instances from the dataset may lead to introduction of the new problem of data imbalance. A highly ranked app may have fewer instances in comparison to a relatively low ranked app, which may have a substantial number of instances associated with it. In other words, this is how frequent usage patterns of apps work. More an app is used by owners, more dataset instances related to that app result. We remove all apps which are less informative in terms of frequency of usage. Low informative apps contribute less in achieving good predictability of a classification model. We define low informative apps as follows.

Definition 4: (Low Informative Apps)
An app \(\mathcal{A}\) is low informative if the number of instances related to \(\mathcal{A}\) in dataset \(D\) is less than a certain minimum cardinality threshold \((\eta)\). It can be calculated as follows.

\[
\text{Low}(\mathcal{A}) = \begin{cases} 
1, & \text{if } |\{T|T.a_i = \mathcal{A}\}| < |D| \times \eta \ 
0, & \text{otherwise,} 
\end{cases}
\]

where, \(T.a_i\) is the value of the attribute \(a_i\) for tuple \(T\) in dataset \(D\). Here, \(a_i\) is the column specific to all the apps name. The value of \(\eta\) is in the range \([0, 1]\). \(|\cdot|\) represents the number of instances or tuples \(T \in D\) related to attribute value \(a_i = \mathcal{A}\).

D. Ranking Scheme

The scheme for ranking is depicted as flowchart in Figure 1. The process starts with the mobile usage records. In every iteration, we remove all app from the dataset. Removing an app means eliminating all the records containing the app. We calculate the PMI of the partially reduced dataset and compare its influence factor. We then normalize the PMI score and use it for ranking the most informative apps in the dataset. Top \(k\) apps are used to create the final reduced dataset by removing all instances from the dataset except records from top apps. We apply a post-processing step on the top list by removing those apps from the list that are associated with very low number of instances in the dataset.

We use the high informatics apps (by removing low informative apps from the ranked list) for data reduction. We use state-of-the-art classifiers for detecting legitimate owners using the reduced dataset for training. The performance of different candidate classifiers applied to our reduced dataset is discussed.

IV. EXPERIMENTAL EVALUATION

This section outlines the assessment designed to validate the proposed ranking method in terms its prediction effectiveness.
A. Data Description

We use two datasets: a private dataset which is called the UCCS dataset, and a public dataset which is known as MIT dataset [7]. Characteristics of the datasets are given in Table I (# read as total number of).

TABLE I: Smartphone usage Dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Users</th>
<th>#Instances</th>
<th>#Features</th>
<th>#Apps</th>
<th>Year of Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCCS</td>
<td>25</td>
<td>118112</td>
<td>20</td>
<td>293</td>
<td>2015</td>
</tr>
<tr>
<td>MIT</td>
<td>50</td>
<td>186240</td>
<td>22</td>
<td>126</td>
<td>2009</td>
</tr>
</tbody>
</table>

B. Results and Analysis

We run experiments using macOS Sierra, on a 2.7 GHz Intel Core i7 processor with 16GB 1600 MHz DDR3 memory. We perform two different experiments based on PMI and weighted PMI based ranking of the apps. For our current work, we consider only top 10 apps. The normalized ranking with the applications’ names (top 10 apps) and PMI score for both datasets, UCCS and MIT are shown in Table II. Table III reports the top 10 apps based on the weighted ranking score.

TABLE II: Top 10 Apps based on PMI based ranking

<table>
<thead>
<tr>
<th>UCCS dataset APPS</th>
<th>Ranking score</th>
<th>MIT dataset APPS</th>
<th>Ranking score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TouchWiz Home</td>
<td>1</td>
<td>Menu</td>
<td>1</td>
</tr>
<tr>
<td>Clash of Clans</td>
<td>0.98374</td>
<td>FExplorer</td>
<td>0.98842</td>
</tr>
<tr>
<td>TwLancher</td>
<td>0.97508</td>
<td>MediaGallery</td>
<td>0.98395</td>
</tr>
<tr>
<td>S Voice</td>
<td>0.95761</td>
<td>mce</td>
<td>0.98081</td>
</tr>
<tr>
<td>Apus</td>
<td>0.95576</td>
<td>profileapp</td>
<td>0.980729</td>
</tr>
<tr>
<td>Block Puzzle Mania</td>
<td>0.95222</td>
<td>Appmngr</td>
<td>0.980725</td>
</tr>
<tr>
<td>Solo Launcher</td>
<td>0.95008</td>
<td>Camera</td>
<td>0.980382</td>
</tr>
<tr>
<td>Sony Ericsson Home</td>
<td>0.949617</td>
<td>Pinboard</td>
<td>0.97884</td>
</tr>
<tr>
<td>Caller</td>
<td>0.949597</td>
<td>gs</td>
<td>0.978773</td>
</tr>
<tr>
<td>Holly Quran</td>
<td>0.949105</td>
<td>VoiceRecorder</td>
<td>0.978298</td>
</tr>
</tbody>
</table>

To assess the prediction effectiveness of the reduced datasets based on proposed ranking schemes, we use state-of-the-art classification models and report Equal Error Rate (EER), and running time of each classifier. EER is the common value at which false acceptance rate and false rejection rate are equal. In other words, it is the rate at which both acceptance and rejection errors are equal [14].

1) Performance of the Classifiers: We use four classifiers; IBK [15], Decision Table [16], J48 [17] and Random Forest [18] using the Waikato Environment for Knowledge Analysis (Weka)1. We report EER and execution time required by the candidate classifiers to process the original datasets in Table IV, using PMI-based ranking in Table V and weighted ranking in Table VI.

Fig. 1: Illustration of App Ranking Scheme

Fig. 2: Size of the reduced dataset after applying PMI ranking and wRanking

Sizes of the reduced datasets after applying PMI ranking and weighted ranking scores are shown in Figure 2. We achieved drastic data reduction using our proposed ranking schemes.

1http://www.cs.waikato.ac.nz/ml/weka/
possible that some features are redundant or irrelevant, and so accuracy is using feature selection. For each dataset, it is cuts down the running time, while producing a better detection rate. On the other hand, the use of PMI drastically that the weighted ranking scheme is effective in enhancing results.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>UCCS EER</th>
<th>MIT EER</th>
<th>UCCS Time</th>
<th>MIT Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBK</td>
<td>14.5%</td>
<td>16.5%</td>
<td>291.942</td>
<td>2280.29</td>
</tr>
<tr>
<td>D.Table</td>
<td>13.9%</td>
<td>11.36%</td>
<td>295.37</td>
<td>1804.25</td>
</tr>
<tr>
<td>J48</td>
<td>15.8%</td>
<td>9.004%</td>
<td>174.626</td>
<td>380.0</td>
</tr>
<tr>
<td>R. Forest</td>
<td>14.99%</td>
<td>10.43%</td>
<td>363.88</td>
<td>851.19</td>
</tr>
</tbody>
</table>

Eliminating the undesirable features can enhance the accuracy. We use Chi-square attribute evaluation [19] to enhance the obtained results using the IBK classifier for all previous experiments. Figure 3 shows all 22 features and the corresponding ranks. In Figure 3, we observe that app_date, app_date.1, app_date.2, loc_ids, frequencyOpen, frequencyClose, month and apps are some of the best features in the dataset.

We repeat our experiments five times and compute the average of EER and time required to run IBK. We use the original datasets and the reduced datasets based on the two kinds of ranking. We achieve better running time and error rate (EER) with the PMI ranking approach. Table VIII shows the improved performance applying ranking and feature selection.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Original EER</th>
<th>MIT EER</th>
<th>Original Time</th>
<th>MIT Time</th>
<th>Original ROC</th>
<th>MIT ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBK</td>
<td>0.924</td>
<td>0.941</td>
<td>0.928</td>
<td>0.895</td>
<td>0.990</td>
<td>0.991</td>
</tr>
<tr>
<td>D.Table</td>
<td>0.960</td>
<td>0.972</td>
<td>0.955</td>
<td></td>
<td>0.990</td>
<td>0.991</td>
</tr>
<tr>
<td>J48</td>
<td>0.954</td>
<td>0.973</td>
<td>0.943</td>
<td>0.981</td>
<td>0.979</td>
<td></td>
</tr>
<tr>
<td>R. Forest</td>
<td>0.982</td>
<td>0.972</td>
<td>0.979</td>
<td>0.998</td>
<td>0.996</td>
<td></td>
</tr>
</tbody>
</table>

Using the reduced dataset in terms of rows and columns, for the IBK classifier, we are able to reduce the EER for the original MIT from 16.5% within 2,280 seconds as shown in Table. IV, to 9.04% EER with 273.91 seconds. This means that although IBK has lower results in the initial experiments, but when we use improved mutual information and feature selection, it produces better results in both EER and time. Figure 4 illustrates the comparison between different ranking approaches for the MIT dataset.

Weighted ranking we reduced the dataset to 62,592 instances to make sure that all the target classes are included in the chosen data subset. Identifying the most informative apps to improve the performance of the classification model is our goal. Product Mutual Information is able to identify top ranked apps and eliminate the rest. We feel that non-ranked apps are contribute little in the identification of the owners and hence eliminated for data reduction. Data imbalance is an issue that may occur during the elimination process. To overcome such situation, we introduced weighted ranking to reduce the effect of imbalance data. The EER of the weighted rank approach is slightly higher than the PMI approach due to the higher number of records and higher number of users. So the EER increases in comparison to the PMI approach.

We also report in Table VII the area under the ROC (AUROC) curve [4] of each candidate classifier for different datasets and compare their effectiveness. Results clearly show that the weighted ranking scheme is effective in enhancing detection rate. On the other hand, the use of PMI drastically cuts down the running time, while producing a better detection rate in comparison to the original dataset.

2) Selecting the relevant features: Another way to improve accuracy is using feature selection. For each dataset, it is possible that some features are redundant or irrelevant, and so...

![Fig. 3: Ranking features using Chi-square test](image-url)
We compare the performance of our scheme with the methods proposed by Li et al. [6] in terms of EER. Our results reported in Figure 5 shows its superiority, by outperforming all other related methods.

V. CONCLUSIONS AND FUTURE WORK

Improving security as well as keeping user’s privacy is very important when authenticating smartphones users. In this paper, we present a scheme to reduce the size of datasets by using a list of most informative apps, with the goal of identifying legitimate owners. Experimental results show that contain app usage records that the overall the proposed scheme substantially reduces the error rate (EER) in detecting legitimate owners of smartphones. We employ feature selection to further improve the performance by removing less important features from the dataset. Distinguishing the primary user of a phone from other legitimate but secondary users will be considered as future work. Some users allow family members to access their devices, so identifying such users also is a goal of future research. Owner identification when installation scenario such as installing new apps and un-installing existing apps will be addressed in our future research.

REFERENCES