Anticipating Application Switching

Xuefeng Hou [1] **Abhishek Kaushik** [1]

Abstract

On the basis of the EMVA dataset, we discuss the feasibility on the topic of Anticipating Application Switching during mobile device interactions. Previous work on the related anticipating application switching is based on the features from mobile device usage logs, sensor data and contextual data, and finished the prediction task mostly with the help of wearable equipment which is in daily mobile phone usage scenarios not necessary. In our study, we focus on the features of user behavior like gaze data and screen touch frequency during the daily interaction with the mobile device. Feature extraction works on with the mobile phone usage features from interactive data, and features of attentive behaviour from the front-facing camera. By OpenFace we process the video and generated the related gaze data. After that we implement a suitable machine learning model for the classification task, evaluate the performance and analyse the generated results. We demonstrate that our method can anticipate the application switching, i.e. which app the user intends to switch, and may help to develop tools or plug-in that increase productivity while minimising possible distractions from the environment. We also discuss about the remaining challenges for improving the performance in the future work. The code is available at the following url: https://github.com/KalvinHou/Anticipating-Application-Switching

1. Introduction

In recent years, there is a rapid increase in the number of smartphone users (Ericsson, 2021). The main reason for this is the multifunctionality and easy accessibility of the smartphones. Because of the mobile applications, one smartphone can serve different purposes like camera, calling, internet, and social media. (AuroraMobile, 2021) On average there are around 60 installed applications in the mobile device of smartphone users. Out of these many installed applications the user only used some set of applications on daily basis. According to a survey (Shen et al., 2021), 76 percent of participants thought that it takes longer time to load the application, and 90 percent of participants ready to use an software that can reduce the loading time of an application. By predicting the next application the user is going to switch to, we can reduce the loading time of an application and can improve the user experience during his interaction with the device.

Figure 1. App recommendations from Android OS

For Example: If a user is currently using Adobe Reader app, and he wants to search some text on Google. By accurately predicting the next application Google, we can reduce the searching time and can increase the efficiency.

With the information of next app, we can preload that app which takes longer time to load like gaming apps and can reduce the loading time of the app.

In our mobile devices we see the set of next probable apps, the number of recommendations (can be 4 or 8) varies from device to device. These recommendations are just based on

[1]University of Stuttgart, Germany. Correspondence to: Xuefeng Hou <st175367@stud.uni-stuttgart.de>, Abhishek Kaushik <st176872@stud.uni-stuttgart.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

the previous application usage history. There are also other important features which can influence the next application like dominant apps, location, status of the battery, screen orientation, touch events, notifications,time window etc... To accurately predict the next application of the user at certain instance we should consider all the above features in our prediction model.

In our project we analysed the influence of different features on the performance of our model to predict the next application, and implemented a well-performed machine learning model for this classification task with the help of all of the features we extracted.

2. Related prior work

In this part,we have summarized the prior work relating to application switching and predicting the user's next application.

Application Switching

There has been very little research done on the application switching behavior of smartphone users. Switching behavior from one application to another may depend on various factors like, current application, mood of the user, time of the day etc. (Turner et al., 2019),in his work using network science the author is able to find a common application switching behavior between different smartphone users irrespective of the number of installed applications, and volume of switches and also found that there is a decrease in the number of switching from important apps on the user's phone to less important apps.

Application Prediction

Predicting the next application the user is going to use has been a challenging research topic for a very long time. Because of the huge number of installed applications on the user's device and large uncertainty in application usage pattern, it is difficult to predict the next application of the user.

In the early work, (Verkasalo, 2009), included the contextual information which are Name of the application (What?), Time of the application used (When?), Location(Where?) from the raw data in the model for the prediction of the next application.

(Shin et al., 2012),used a total of 37 features collected from various sensor data, such as location, bluetooth, wifi status, screen orientation, 3D accelerometer, and so on. They used a Greedy Thick Thinning algorithm for feature selection to incorporate the important features in their model. They employed the Naive Bayes Classifier in their app model, which is a probabilistic model, to classify applications.

With the increase in the popularity of neural networks, and advances in the hardware to train the large neural networks. The performance of Recurrent Neural Network (RNNs) and Long Short Term Memory (LSTMs) in time sequence data is promising.(Xu et al., 2020) used LSTM to predict the next application the user is going to switch to. Because of the ability of LSTM to capture the temporal sequence of Application usage data, this model outperforms the previous work. In this work they implemented Softmax activation instead of Sigmoid activation to predict the top-K applications.

In recent studies, (Shen et al., 2021) employed deep reinforcement learning to predict the next application the user is going to switch to. For 21 days, data from over 400 smartphone users was collected from a major city's mobile carrier. They trained a general agent for all users and then trained a user specific agent on that specific user app usage data. The name of the app, the location (where the app is used), and the time (at what time that app is used) are used as the features to train the model. By this work they are able to outperform all the recent works in predicting the next application.

3. Dataset

3.1. Dataset introduction

We have used the EMVA (Everyday Mobile Visual Attention) (Bâce et al., 2020) dataset in our whole project. EMVA dataset is unique in nature because it collects not only the mobile usage logs of the user but also the videos from the front camera of the device. The data collection approach in this dataset is different from previous application usage datasets. Previous application usage dataset was collected from the mobile carrier of a city from different towers. data in the EMVA dataset is collected from the different participants for over two weeks using a data collection app, which is available on the google play store. (Bâce et al., 2020) EMVA dataset consists of metadata like mobile device usage logs, sensor data, location data, and bluetooth data. In this dataset the device usage logs data is important for our task to predict the next application. In device usage logs we have application data, touch events(when it happens not where on the screen), screen orientation data, wifi connectivity state. The activity of the user is also recorded in this dataset while the user is using its mobile device. We also have notifications data in this dataset, which have the information of the package name and also the status of the notifications (appeared or removed).

In the EMVA dataset (Bâce et al., 2020) the videos are recorded from the front facing camera of the mobile device. There is in total around 472 hours of video for all the 32 participants. With the video data, the dataset is able to capture the true behavior of the user while interacting with the device. We can extract useful information from this video data like gaze data, eye contact detection, and emotion

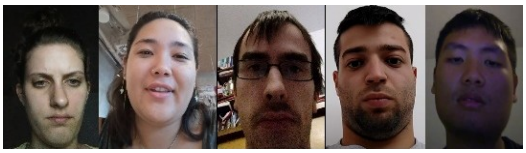


Figure 2. Sample images from the video data

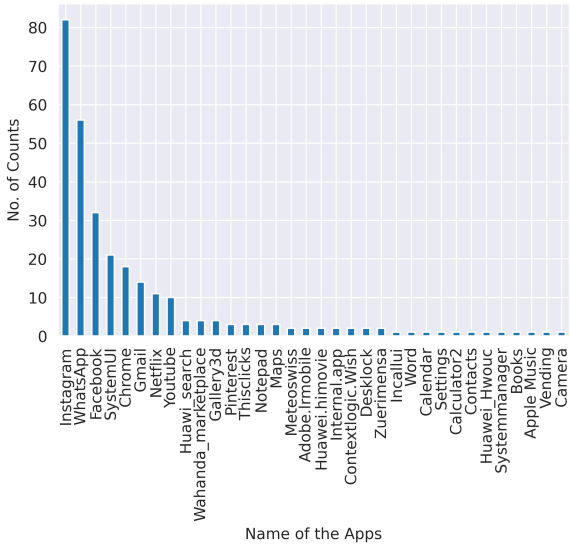


Figure 3. Frequency of the apps

of the user.

3.2. Data Analysis

To get some basic understanding of the EMVA dataset, we did the data analysis on different parts of the dataset.For our analysis, we randomly chose the 5 out of 32 participants. Based on our findings from this analysis, we used some feature in our final prediction model. For these 5 participants we are also able to generate the gaze data from the video data of EMVA dataset.

Application Data

In the application data we have the name of the application running in the foreground, and when the application is running (date and time) with the timestamp.

To predict the next application the user is going to switch to, the frequency of the applications is an important information that we can use in our prediction model. There are dominant applications which are used by the user on a daily basis and also there are less dominant applications that are used by the user once or twice per two weeks. The count of dominant applications is very few as compared to the less dominant applications. We found a similar type of application usage pattern in all 5 participants. From the

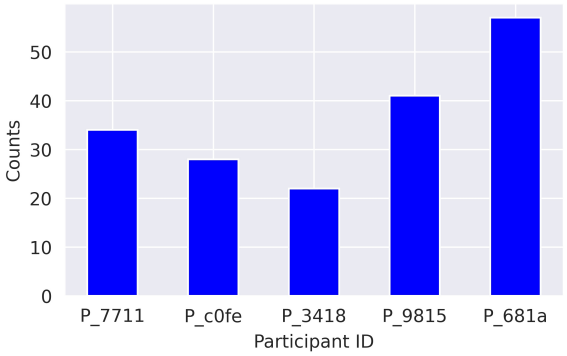


Figure 4. Number of Unique apps used by the users

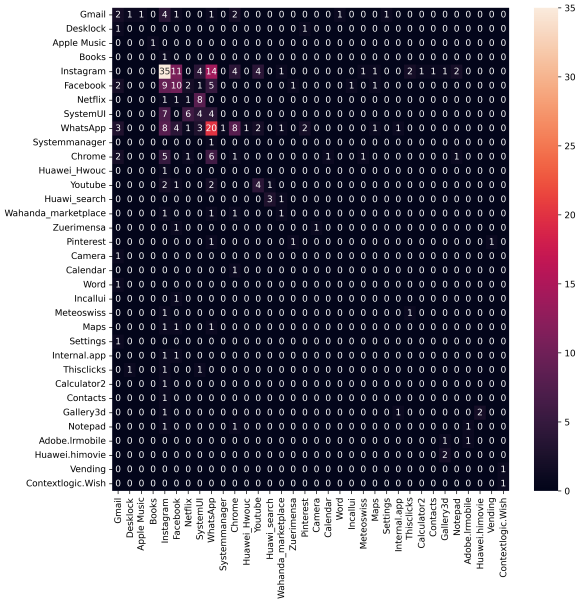


Figure 5. Transition Matrix

above findings, we can say that there is a high probability of switching to dominant apps rather than less dominant apps. We found the common dominant apps (Popular apps like: Whatsapp, Facebook, Youtube, Instagram ,etc..) in all the 5 participants application usage data, although the most frequent app is different for all the participants.

During the data collection period, the overall number of distinct applications used by various users varies dramatically. There are many applications which the users has only used once in the data collection period. As the number of unique applications increases the accuracy to predict the next application decreases.

By transition matrix we want to see the influence of the current application on the next application used by the user. Transition matrix is a measure to find how many times user switches from one app to another apps. There are many

	name	time_x	timeDate	event	package	category
231	NaN	NaN	10.08.2019 15:36:32.420	posted	com.whatsapp	None
232	NaN	NaN	10.08.2019 15:36:32.421	posted	com.whatsapp	msg
233	com.huawei.android.launcher	4.728819e+13	10.08.2019 15:36:33.353	NaN	NaN	NaN
234	com.whatsapp	4.728922e+13	10.08.2019 15:36:34.381	NaN	NaN	NaN
235	NaN	NaN	10.08.2019 15:36:34.518	removed	com.whatsapp	msg

Figure 6. Showing influence of Notifications on Application switching

transitions within a set of applications as compared to the rest of the applications. From the figure 5 we can see that there are many number of transitions between Instagram, Facebook, Chrome, Whatsapp and Gmail.These are also the frequent apps used by this participant,it appears that there is some kind of grouping between some of the applications used by the user. Except from these set of applications there are not many transitions between rest of the other applications.Other participants also followed a similar application usage pattern. The current application of the user is an important feature to predict the application switching. We tested our model by using the current application as only feature to predict the next application user going to switch to.

Notifications Data

Smartphone users receive many number of notifications from different applications on daily basis. Because of distraction from notifications many users don't like to get notifications. In our analysis also out of 5 participants 2 participants have blocked the notifications. We only have notifications data for rest of the 3 participants.The two participants got the most number of notifications from the Whatsapp(Messenger app).

As shown in figure6,after merging the application and notification data of 1 participant, there is a pattern in switching to that application from which the participant got the notification.The switching time to the application depends on the package name of the notification from which the user has received the notification.

Similar pattern from getting notification form an application and then switching it to that application, also found with other participants.

4. Feature extraction

4.1. Feature extraction from video data

4.1.1. Video features extraction

To extract features from the video data we created the gaze data with the help of OpenFace (Baltrusaitis et al., 2018). OpenFace is a state of the art tool to generate facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation. From the gaze data we used the gaze direction vector, and also the eye gaze direction as features in our prediction model. To generate the gaze data, one challenge is to correct the orientation of the videos from the EMVA dataset (Bâce et al., 2020) before feeding it to OpenFace for further processing. From correcting the orientation of a video to generating the gaze data it takes approx 30 minutes for a 15 minute video. There are around 14000 videos for 32 participants which is very time consuming. So, we limit our analysis to 5 participants because of the time constraint.

4.1.2. Video features processing

As for the raw video features after extraction by OpenFace, it's necessary to process the data in a particular time window. The reason why we used a time windowing method is that we want to predict the application switching before it happens, so it would be important to focus on the time period in the previous t seconds, t can be 1/3/5. Not specifically for the raw video features, all other necessary features which we want to extract from the interactive data should also follow this limitation, in order to get the most relevant features.

After that, we manually selected gaze data and removed eye landmark which we thought having stronger connection with the user behaviors and mobile phone using habits. As Figure 7 showed, each feature represents one coordinate value of the gaze data. We calculated the mean, variance and standard deviation for each feature, and collected $3 \times 8 = 24$ features for the video data.

4.2. Feature extraction from interactive data

From the interactive data (the user log from the mobile phone), normally we could get several useful features. For example, *Application Data* recorded which app the user opened at a certain timestamp; *Device data* recorded the device status changing like *Battery charging*, *Brightness changing*; *Touch Data* recorded when the user touched the mobile phone screen. With all the user log files, we managed to divide these features into two classes: *Switching related features* and *Time windowing features*.

The overview of the different features explored in this work is provided by the Table 1. In details, the categorical method of the app is based on apple app store and we did some minor change, like combining less used and quite similar app categories into one to make it less complicated. The Switching time of the day follow the format as *HH/mm/ss*.

So far, we extracted $24 + 7 = 31$ features from processed video and user log data, however so far they are still unprocessed invalid data which we could not use directly. So it's

	confidence	success	gaze_0_x	gaze_0_y	gaze_0_z	gaze_1_x	gaze_1_y	gaze_1_z	gaze_angle_x	gaze_angle_y	...	eye_lmk_Z_46	eye_lmk_Z_47	e
0	0.98	1	-0.058954	-0.457938	-0.887027	-0.135234	-0.456709	-0.879277	-0.110	-0.478	...	416.0	416.9	
1	0.98	1	-0.053291	-0.443624	-0.894627	-0.143240	-0.442300	-0.885355	-0.110	-0.462	...	426.4	427.4	
2	0.98	1	-0.074696	-0.357094	-0.931077	-0.129029	-0.425962	-0.895493	-0.111	-0.405	...	423.7	424.7	
3	0.93	1	-0.042742	-0.400226	-0.915419	-0.191986	-0.352971	-0.915725	-0.127	-0.390	...	435.4	436.6	
4	0.03	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000	0.000	...	369.3	370.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	

Figure 7. Overview of the different features explored from processed video

Class	Features
Switching-related	name of the current app
	using frequency of the current app
	using duration of the current app
	category of the current app
	switching time of the day
Time-windowing	device status changing
	screen touch frequency

Table 1. Overview of the different features explored from user log data

necessary to process the data and make the features valid.

5. Data Preprocessing

In order to get the valid application switching, we need to figure out the how the user operated it. When the user wants to switch the app from A to B, then it's possible for him to close the current app and open the home page for selecting the new app. So actually app A to home to B is not what he wants, which it should be directly from A to B; Furthermore, the EMVA dataset is recorded by an app called *VisualAttentionRecorder*, so any switching including this app is invalid.

The recording duration of each video session is maximum 15 min, within the duration of video recording the participant can switch several apps as he wants or keep using one app from the beginning to the end, which means it's possible to extract no valid application switching within one video session. In addition it's important to focus on the invalid switching situation, which the current app is at the last position from the previous video session, while the switched app is at the first position from the next video session. After calculating the difference of timestamp we can define whether it belongs to a valid data. If the difference of timestamp is less than 1 min (including the time which the user closed the *VideoAttentionRecorder* and reopened it), then we regarded it as a valid application switching.

6. Task Modelling

6.1. Classifier selection

Considering the difficulties of collecting large amount of data, training a neural-network-based model for our case would be unrealistic. After feature extraction, the valid samples for each participants are from 100 to 400, therefore traditional machine learning model would be a better choice, like (Mucherino et al., 2009) kNN, (Breiman, 2001) Random Forest, (Li & Jain, 2009) AdaBoost and other ensemble learning methods.

While doing data analysis, we discussed about the importance of the using frequency of apps, since the user intends to use particular apps much more often than others, which we commit as the user-specific mobile phone using habits. Therefore it comes to an assumption, what if we predict the result of application switching by the most frequent app which the user used before? Is this possible to get to know about the user behavior and mobile phone usage pattern by simply recommending the app which the user used frequently? In order to discover the answer and make a more detailed comparison, we added *Dummy Classifier* with the method *the most frequent* and regarded it as an evaluation baseline, for further analysis and comparison towards the performance of our training model.

6.2. Metrics specification

As we all can find on our own mobile phone, as it shows in Figure 1, when it comes to the application recommendation from the system, multiple apps would be displayed instead of only one. In iOS system it comes to 4 and 8, in android system it comes to 5 and 10. Which means it would be difficult to predict the exact 1 app that the user intend to switch, so giving more suggestions could help to improve the final performance. Therefore we also added the Top-K accuracy into our metric for the evaluation part, in order to train a more practical model for daily life.

6.3. Feature selection

Since we have totally 31 features, containing 24 features from the video data and 7 features from the interactive data, therefore a necessary feature selection could help to find the most effective features for classification recognition from this large number of features, so as to achieve the compression of the dimensionality of the feature space, i.e. to obtain a set of "fewer and more precise" features with a lower classification error.

As we all know that feature selection methods can be essentially divided into wrappers, filters and embedded methods. On account of getting the better result directly from the training model and improve the prediction performance, embedded methods combines the feature selection with the process of training which is basically based on the machine learning model, and it could select the most relevant features for our specific task.

For the well-known L1-norm (LASSO) feature selection (Tibshirani, 1996), its objective is to minimize the loss while combining the L1-constraint into the loss function. However it is restricted to the linear function which is quite unsuitable for our case. Besides L1-norm needs a large amount of samples for converging the final weights of each feature, which is impossible for the limited samples we got from EMVA dataset.

In contrast, tree-based feature selection with random forest has the advantages of high accuracy, robustness and ease of use, moreover it perfectly matches our model and our task. By calculating the mean decrease accuracy of each feature, which shows the influence towards the whole accuracy, we could select the more valuable features from all of the 31 features.

7. Performance of the training model

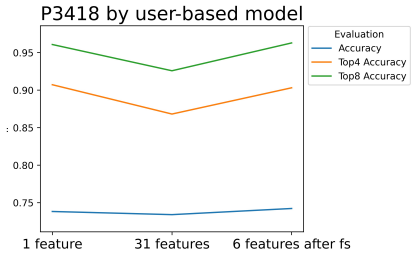
7.1. User-specific model performance

	Accuracy	Top4 accuracy	Top8 accuracy
kNN	0.2995	0.5645	0.6291
RF	0.4291	0.6322	0.7484
AdaBoost	0.3711	0.5645	0.6936
Dummy	0.2782	0.2782	0.2782

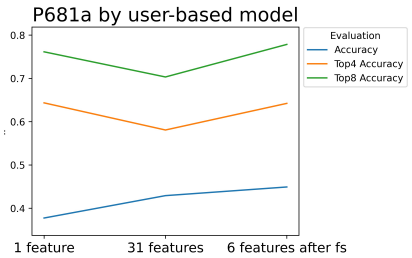
Table 2. Model comparison after feature selection in P681a

	Accuracy	Top4 accuracy	Top8 accuracy
kNN	0.7320	0.8453	0.9278
RF	0.7422	0.9031	0.9629
AdaBoost	0.7320	0.8763	0.9691
Dummy	0.3924	0.3924	0.3924

Table 3. Model comparison after feature selection in P3418



(a) Participant 3418



(b) Participant 681a

Figure 8. Accuracy improves after feature selection

Due to the limited storage and time, we processed 5 participants with complete valid features. Without the help of feature selection, we firstly put all the 31 features together to train each of the participant with their own samples. Here we show two participants as examples which from Table 2 and Table 3.

For comparing three basic machine learning models from all of our participants, we find that only *Random Forest* remains a higher stability than others, which means *Random Forest* no matter for which participant tends to have a higher accuracy than the average value. On the contrary, the remaining two models certainly have some limitation when dealing with such classification problem, and create outliers on particular situations. For example from Table 2, *kNN* has obviously much lower accuracy with only around 30% than other two models. Therefore we choose *Random Forest* as our training model for the following evaluation.

These two participants are very representative that participant 3418 from Table 3 has a much higher accuracy. When we evaluate the top-k accuracy, it can be around 90% for such a model, so there have to be some mobile phone using habits being found by our training model for this participant. However when we look at the participant 681a from table 2, the results are in contrast quite poor. Average 35% accuracy can be achieved when predicting the next app which the user intends to switch. With top-4 accuracy and top-8 accuracy it increases into around 55% and 66%. But if we compare to our baseline *DummyClassifier* we can find that our model still find some usage patterns for around 10% higher than the baseline.

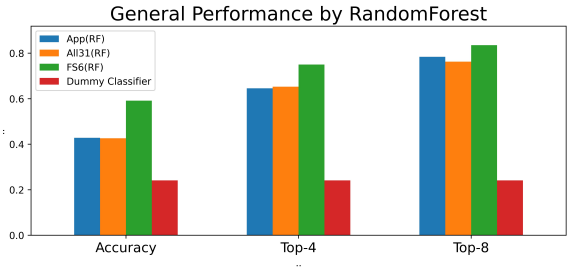


Figure 9. General prediction performance

As the results from these two representative participants show, there are large uncertainty in user behavior and mobile phone usage pattern, each individual behaves differently in different using situations, and has much various thought of what to switch or even what to do. Furthermore, different participants may tend to use their phones in their specific situations and locations, which may influence the video features extraction on light condition, angle limitation, face direction etc. So the result after training the same user-specific model for different participants varies a lot. But even the worst participant we got still has a higher single accuracy than the baseline, not to mention if we randomly select one app among over 60 apps as the final prediction, the random accuracy can only be less than 2%.

7.2. General model performance

We grouped all the participants and collected 1133 valid application switching samples for training a general model as it shows on Figure 9 by the green bar. In contrast to the previous two models, respectively single-feature model and all-features model, our model achieves a higher accuracy for all the metrics. In details, our final general model could reach the accuracy nearly 60% for all of our 1000+ samples from the Table 4, and it gets over 75% when predicting the top-4 possible apps which the user intends to switch, by top-8 it increases to 83.5%.

After adding 30 features to the first single-feature model, the accuracy didn't have much difference and improvement, even for the top-8 metric make it decreased. The reason we think of is that among all the features they are equally weighted, and it can't be chosen manually for the more relevant features. So some of the features improve the prediction performance while some others make the accuracy even worse. It leads us to the step feature selection. With the help of tree-based feature selection, the final results improve a lot and select the features which truly relevant and meaningful.

As we discussed before, the mobile phone system usually gives the app recommendation for at least 4 or 5, when the user clicks *more* button then it increases to 8 or 10.

	Accuracy	Top4 accuracy	Top8 accuracy
Single-feature	0.4282	0.6458	0.7838
All-features	0.4261	0.6528	0.7627
Final model	0.5915	0.7511	0.8352
Dummy	0.2409	0.2409	0.2409

Table 4. General prediction performance by RF

So comparing to these, with the model we processed the accuracy of anticipating the true application switching could be 83.5% overall.

7.3. Generality of the usage pattern

In order to test the performance of our final model towards different individuals, we used the trained model to predict each participant to compare the user-specific model and general model. From Figure 10 we can see, for both participants the prediction performances from the general model are much worse than the its own user-specific model.

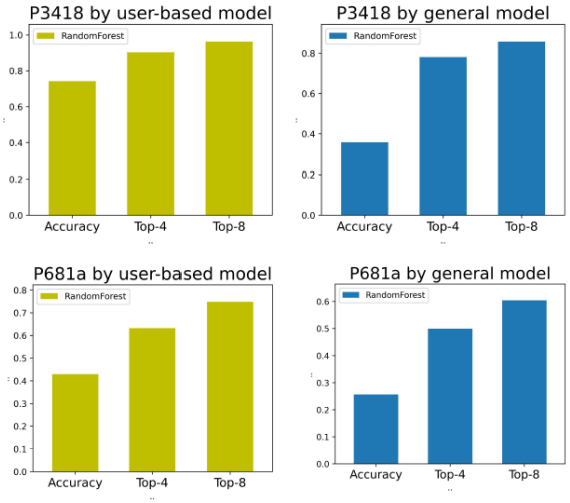


Figure 10. Comparison to user-specific model

8. Conclusions

We think that specific feature selection could be one of the main reason. Previous tree-based feature selection is based on the general model which contains all the participants and over 1133 samples, so the features it selected were actually based on the specific samples, which means those features maybe relevant and meaningful for the general model, but it doesn't mean that they have the same effect and influence to the specific participant.

Furthermore, considering about the varying mobile phone using habits that each individual has, it is also unsuitable for predicting a single individual by the pre-trained model

as a general model, which will get poor results from that.

So far the study described here only focus on the specific dataset (EMVA), and we cannot tell how well the results might be if we transform the model to other similar dataset. However we think the way EMVA dataset collects the samples is not complicated and easy to access, which could be a possible way to wrap the model and data collector as an app or a plug-in for the mobile phone, anticipating the application switching while keep collecting the useful and necessary data for further more training.

The key challenges we faced so far is the difficulties when extracting the features from the raw video of the front camera and from the interactive data of the user log. We did lots of processing for the data in order to make it valid for the training part. In addition the limited samples certainly influenced the performance of training. We have to give the thought of neural network up and turn to solve the problem with traditional machine learning model. With more valid samples from the dataset we may achieve better results by trying with other neural network models. In general, there are large uncertainty in user behavior and mobile phone usage pattern, therefore focusing on the user-specific model for different individual user could get a better performance for such similar classification task.

As for the contribution we did during the whole process of the project, we processed the raw video from the front camera and extracted the valid gaze data as attentive behavior. We also extracted the mobile phone usage features from interactive data. Furthermore we constructed and implemented the training model to classify which application users are likely to switch to and reached better results with our more achievable and accessible model (without other wearable equipment).

9. Future Work

For this task, it mainly focus on the corresponding way of collecting the data but can be practically used among a wide range of applications in real life.

As for the feature extraction from the front-facing camera, we could try with face landmark combining with gaze data to see if there anything interesting happens. Besides, it's also meaningful to define the raw data of the video part since they are just coordinate values. By using other libraries may help to classify the facial expression of each frame into different emotion category, i.e. adding emotion recognition as another key features for the training.

In addition, we got limited dataset when dealing with the problem. In order to process the data and make it much more than before, we could find some methods to split the data. For example, we can regard timestamp as a feature which is contained within an app switching, which means we accept the situation that the user switches app from A to A itself as keep using this app. However it will lead to imbalanced data which contains more non-switching than switching situations. Furthermore generate more artificial data by GAN could also be an option as the data augmentation method which is worth to try.

With more samples the choice of training model could be various. With LSTM model we could put a reasonable feature into our model, which is the past sequence of apps. The current app may truly important to the possible happening switching but the past sequence of apps may be more powerful since it contains the current app and the connection of switching.

References

AuroraMobile. Average number of apps installed by mobile users in china from 3rd quarter 2019 to 2nd quarter 2021. website, 2021. https://www.statista.com/statistics/1239623/china-average-number-of-apps-on-smartphones.

Baltrusaitis, T., Zadeh, A., Lim, Y. C., and Morency, L.-P. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pp. 59–66, 2018. doi: 10.1109/FG.2018.00019.

Breiman, L. Random forests. *Machine Learning*, 45 (1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Bâce, M., Staal, S., and Bulling, A. Quantification of users' visual attention during everyday mobile device interactions. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2020. doi: 10.1145/3313831.3376449.

Ericsson. Number of smartphone users from 2016 to 2021, 2021. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide.

Li, S. Z. and Jain, A. (eds.). *AdaBoost*, pp. 9–9. Springer US, Boston, MA, 2009. ISBN 978-0-387-73003-5. doi: 10.1007/978-0-387-73003-5_825. URL https://doi.org/10.1007/978-0-387-73003-5_825.

Mucherino, A., Papajorgji, P. J., and Pardalos, P. M. *k-Nearest Neighbor Classification*, pp. 83–106. Springer New York, New York, NY, 2009. ISBN 978-0-387-88615-2. doi: 10.1007/978-0-387-88615-2_4. URL https://doi.org/10.1007/978-0-387-88615-2_4.

Shen, Z., Yang, K., Xi, Z., Zou, J., and Du, W. Deepapp: A deep reinforcement learning framework for mobile application usage prediction. *IEEE Transactions on Mobile Computing*, PP:1–1, 06 2021. doi: 10.1109/TMC.2021.3093619.

Shin, C., Hong, J.-H., and Dey, A. Understanding and prediction of mobile application usage for smart phones. pp. 173–182, 09 2012. doi: 10.1145/2370216.2370243.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Turner, L. D., Whitaker, R. M., Allen, S. M., Linden, D. E. J., Tu, K., Li, J., and Towsley, D. Evidence to support common application switching behaviour on smartphones. *Royal Society Open Science*, 6 (3):190018, 2019. doi: 10.1098/rsos.190018. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsos.190018.

Verkasalo, H. Contextual patterns in mobile service usage. *Personal and Ubiquitous Computing*, 13:331–342, 06 2009. doi: 10.1007/s00779-008-0197-0.

Xu, S., Li, W., Zhang, X., Gao, S., Zhan, T., and Lu, S. Predicting and recommending the next smartphone apps based on recurrent neural network. *CCF Transactions on Pervasive Computing and Interaction*, 2(4): 314–328, Dec 2020. ISSN 2524-5228. doi: 10.1007/s42486-020-00045-z. URL https://doi.org/10.1007/s42486-020-00045-z.