Prediction of the next Phone Application Category

A model that companies can use

Jorina Scherff
STUDENT NUMBER: 1273703

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Dr. A.T. Hendrickson
Dr. G. Cassani

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
December 2019

Preface

Over the last few months, I wrote this thesis about phone usage behavior, trying to make a model that companies can use to advertise for their app. To be honest, I don’t like it when people spend more time on their phones than they do talking to each other in real life. I hope this research can help by simplifying which app to choose next and thus causes people to spend less time on their phones. If not, I hope the technology will also be used to send messages like “you already spent a long time on your phone, go for a walk!”.

I enjoyed doing research for this thesis, especially the programming parts and the designing of tables and figures to be able to compare the results of the different models. It was nice to bring the theory we learned last year into practice in a real-world dataset. I will thank dr. Hendrickson and dr. Cassani, the first and second reader, for their help by the process of coming up with research questions and leading me into the right direction. I will also thank my fellow students who were in the thesis meetings for their questions and help. Finally, I want to thank my family and friends for their support.

Abstract

Due to the expanding number of mobile phone applications, consumers have a complex decision process of which app to use. By making predictions of which application someone will use next based on past behavior, a lot of research has tried to make automated recommendations that help the user simplify the process. However, app owners can use such prediction models as well, by deciding when they can send an advertisement banner for their app. Instead of predicting the next app as accurately as possible by using as much information as possible, this model will predict the next app *category* by using only data that is accessible for companies. This has, to our knowledge, not been investigated earlier, so the research question of this thesis was as follows: ***To what extent can the phone application category that someone will open be predicted, while using only mobile phone data that is accessible for companies?*** A dataset is used containing mobile phone usage data with categories assigned to the apps. Different classification models are compared and our findings demonstrate that Support Vector Machine worked best with the features previous app opened, notification, hour of day, and duration of previous app. However, there was a large difference in the recall values of the different categories, mostly caused by the difference in the amount of presence in the dataset. Therefore, it depends on the popularity of the app category how useful this model is.

Keywords: phone use behavior, application category, multiclassification model

Contents

Preface **1**

Abstract **2**

Contents **3**

1. Introduction **5**

1.1. Context 5

1.2. Research Questions 5

1.3. Findings 6

1.4. Outline 6

2. Related Work **7**

2.1. Research to the prediction of the next phone application 7

2.2. Comparison 8

2.3. Choice of techniques 9

3. Experimental Setup **10**

3.1. Data 10

3.1.1. Description of the datasets 10

3.1.2. Preprocessing 10

3.2. Models 13

3.2.1. Logistic Regression (LR) 13

3.2.2. Naïve Bayes (NB) 14

3.2.3. Random Forest (RF) 15

3.2.4. Support Vector Machine (SVM) 16

3.3. Evaluation methods 17

4. Results **19**

4.1. Performances of the different models 19

4.2. Comparison of methods 20

4.3. Input features 21

4.4. Additional performance information per category 21

5. Discussion 24

5.1. Findings 24

5.2. Drawbacks and suggestions on further research 26

6. Conclusion 28

References 29

Appendix I: Information of the phone application categories 33

Appendix II: Results different parameters 34

Appendix III: Results different features tested 36

Appendix IV: Results per category 37

Appendix V: Python code 41

1. Introduction

1.1. Context

Nowadays, smartphones have become indispensable personal gadgets and are used in almost every aspect of people's lives (Cao & Lin, 2017). The expanding number of mobile phone applications (apps) support this, but are also causing an increasingly complex consumer decision process of which app to use. Therefore, the prediction of smartphone apps usage patterns is rapidly growing in importance (Xu, et al., 2013). When the technology knows in which category the next app will belong, automated recommendations for the next app can be displayed that help the user simplify the process (Shin, Hong, & Dey, 2012). Other advantages for the user are the improvement of the device usability, reduction of load time of apps, and cached network content which leads to much higher network speeds (Shin et al., 2012). Companies can profit of such a model as well in their advertising strategies. For example, if the algorithm predicts that the next app category will be *Communication*, Whatsapp can send a banner to the phone screen. This is a special form of targeted advertising, named online behavioral advertising (Smit, Noort, & Voorveld, 2014), and results in a higher chance that the user will choose that one.

Receiving a banner for an app can help to simplify the process of choosing the next app, but companies need to know when they can send this banner. Therefore, the prediction of an app category is important: if companies know when the user is likely to be going to use an app of a specific category, a company having an app in that category can advertise for it. A lot of research is already done, several techniques are tried and different features are used to predict the next phone application. However, the focus of these models was mostly on predicting the next app as accurately as possible, and therefore using as much information available as possible. To make a model that companies can use, only features that companies do have access to can be considered as inputs. Moreover, in this case, the next app *category* that someone will open has to be predicted, instead of the next app itself, so that companies can use this information in their advertising strategy.

1.2. Research Questions

To investigate if it is possible to build such a model, the research question of this thesis is as follows: ***To what extent can the phone application category that someone will open be predicted, while using only mobile phone data that is accessible for companies?***

For this research, a dataset containing mobile phone usage data was available, including the division of the applications into different categories. Using this dataset, the following sub questions are answered to be able to answer the main question:

1. *Which model predicts the next phone application category best?*

Logistic Regression, Naïve Bayes, Random Forest and Support Vector Machine models are built and compared, using the following input features: time (in hours), previous app opened, duration of previous app, and notification.

2. *Which features are useful in the prediction of the next phone application category?*

All combinations of the four features are tested in the best performing model to see if there is any difference in performance.

3. *Is there any difference in the predictability of the different categories?*

The recall of every category is calculated to see if there is a difference between categories. Moreover, to see which categories are often mixed up, confusion matrices are made as well.

Because there are 44 different application categories, with one occurring more often than another, a division is made in all questions between five different prediction problems: top 1, top 3, top 5, top 10 and all categories. For example, the category *Communication* occurred most often, so for top 1, this category is predicted versus all other categories. This is done to be able to investigate the extent to which the next app category can be predicted.

1.3. Findings

The research showed that a Support Vector Machine works best for this problem, using the four features previous app opened, hour, notification, and duration of previous app, although this last one didn't influence the accuracy. The model had an accuracy of around 55 percent, but there was a large difference in the recall values of the different categories. The more different categories that needed to be predicted, the more the most occurring category was predicted, which leaded to a less accurate classification model. Therefore, it depends on the amount of appearances of a category how useful this model is and to what extent this category can be predicted.

1.4. Outline

First, related work is described that contains relevant information that is gathered by other researchers in this field. Then, the choice for models is explained and the experimental setup describes the theoretical working of these models with their programming parts, as well as the needed preprocessing work. After that, results are given in tables and figures and these are shortly explained. A discussion is added afterwards which combines the results with the knowledge gathered from other research. Also the drawbacks of this research are mentioned in this section, as well as suggestions for further research. At the end, a conclusion gives answers to all research questions.

2. Related Work

The prediction of app usage refers to the task of predicting the next app that will be opened by a given user at a given time (Cao & Lin, 2017). Because of the importance of this prediction, a lot of research is already done in this field, several techniques are tried and different features are used as inputs for models to come up with an accurate prediction. This section describes these different models first, and uses another research for the comparison of these models. At the end, a choice is made for the models that are built for this specific research.

2.1. Research to the prediction of the next phone application

Several researchers investigated the prediction of the next app by using different inputs in the models. Böhmer, Hecht, Schöning, Krüger, and Bauer (2011) found that there is a difference in app categories during the day; news applications are most popular in the morning, games at night and apps in the category of communication dominate through most of the day. Shin, Hong, and Dey (2012) agree with this and also found with contextual information in a smartphone that, among other things, last application opened and cell ID are important influences. They made a Naïve Bayes model with much more features, like GPS information and Wi-Fi status, to predict the ten most probable apps and had an accuracy around 88 percent. The correlation between sequentially used apps also has a major contribution on the prediction accuracy according to Huang, Zhang, Ma, and Chen (2012). Xu et al. (2013) made a model that took into account a lot of inputs out of three key everyday factors: contextual signals, community behavior and user-specific preferences and history, whereas Do and Gatica-Perez (2014) predicted the application someone will use based only on the current context consisting of location, time, app usage, Bluetooth proximity, and communication logs.

Not only are different features tested, but several techniques are used as well. Tan, Liu, Chen, and Xiong (2012) used a Prediction Algorithm with Fixed Cycle Length; Liao, Pan, Peng, and Lei (2013) proposed a Temporal-based Apps Predictor made with MaxProb and MinEntropy using a global, a temporal and a periodical usage feature to dynamically select the *k* apps with highest probability to be used next; and while a lot of models used real-life mobile phone data that may contain noisy instances, Sarker (2019) presented a robust prediction model for real-life mobile phone data for individual users avoiding this problem by first identifying and eliminating the noisy instances using a Naïve Bayes classifier and a Laplace estimator, and then making a prediction model using a Decision Tree. Another kind of approach is done by Lim, Secci, Tabourier and Tebbani (2016). They proposed data clustering techniques to create usage clusters of mobile applications via aggregation of similar profiles, and showed that these classes can be utilized to analyze and predict future usages of each app (Lim, Secci, Tabourier,

& Tebbani, 2016). The research of Baeza-Yates, Jiang, Silvestri, and Harrison (2015) is done with the intention to fasten the searching process of the app someone will use, by making a prediction mechanism that allows to show which app the user is going to use in the immediate future. The authors used the Parallel Tree Augmented Naive Bayesian Network (PTAN) for classification as their prediction technique, and the following features were taken into account: basic features that were directly obtained from sensors of mobile devices (time, latitude, longitude, speed, GPS accuracy, context trigger, context pulled, charge cable, audio cable), and session features (sequentially used apps, using word2vec). They got a precision of around ninety percent. Zhu, Cao, Chen, Xiong, and Tian (2014) proposed an approach to enrich the contextual information of mobile apps to better classify these apps into some predefined categories, by exploiting the additional Web knowledge from the Web search engine and extracting some contextual features from the device logs of mobile users. All information is combined into a Maximum Entropy model for training the mobile app classifier (Zhu, Chen, Xiong, Cao, & Tian, 2014).

2.2. Comparison

Because so many different studies have been done on mining smartphone data for uncovering app usage patterns, Cao and Lin (2017) surveyed these existing studies and summarized their differences and similarities. They found that the features used in smartphone data mining are typically categorized into explicit and implicit types, where explicit features are readily available information extracted from the phone, and implicit features are subtle derived app statistics (Cao & Lin, 2017). More precisely, the explicit features can be split into four types: location, time-based, phone settings, and phone status features; and the implicit features require feature engineering and data modeling.

It is proven that implicit features are highly effective in app usage predictions, but their performance improves when adding explicit features (Cao & Lin, 2017). In several works, a large amount of different types of sensor readings is collected, such as time, GPS and app usage. However, more data is not always better, as not only the energy and storage consumption are larger, but there is also a higher chance of noise data (Liao, Pan, Peng, & Lei, 2013). Therefore, it is promising to use only data that companies do have access to, and this study uses the following four features: hour of the day, previous app opened, duration of previous app opened and whether a notification was send or not. Hour of day is an explicit feature, while previous app opened, duration of previous app and notification are all implicit features.

The main difference between this research and earlier research is that the focus of other models was mostly on predicting the next app (category) as accurate as possible, and therefore using as much information available as possible, whereas this new research focuses on making a model that companies can use. So, only features that companies do have access to are considered as input features, and instead of predicting the next app, the category will be predicted so that companies can advertise at the right time for their app.

2.3. Choice of techniques

In the area of mining mobile phone data, association learning and classification learning are the most common techniques to build a rule-based user behavior model (Sarker, 2019). Association learning searches for relationships between variables, whereas classification learning generalizes a known structure to apply to new data. As association learning produces a large number of redundant rules, making the prediction model more complex and ineffective (Sarker, 2019), classification learning is used in this research, which tries to find rules for a labeled training set, to learn a model that maps unlabeled instances to class labels (Becker, Kohavi, & Sommerfield, 2001). Machine Learning is suitable for exploiting correlations between multiple variables (Kamisaka, Muramatsu, Yokoyama, & Iwamoto, 2009) and several methods exist for classification. A big division within this group is between discriminative and generative models, where discriminative classifiers directly map the inputs to the class labels and generative classifiers learn a model of the joint probabilities of the inputs and the labels (Ng & Jordan, 2002). Discriminative classifiers are almost always preferred above generative ones, but also have disadvantages, like the lack of elegance and being like black-boxes (Srihari, 2018), and Ng and Jordan (2002) showed that generative models are sometimes better; it depends mostly on the training size. It is interesting to compare these different learning methods. Logistic Regression is the oldest and one of the most common approaches to solve the classification task (Memisevic, Zach, Hinton, & Pollefeys, 2010), whereas Naïve Bayes is used several times in the classification of mobile phone data (Shin, Hong, & Dey, 2012; Baeza-Yates, Jiang, Silvestri, & Harrison, 2015; Sarker, 2019). As Naïve Bayes and Logistic Regression form a generative-discriminative pair for classification (Ng & Jordan, 2002), these models are both made.

Another big division within the classifiers is between linear and non-linear models. Naïve Bayes and Logistic Regression are both linear methods, so two non-linear methods will be made as well. Sarker (2019) showed the effectiveness of a Decision Tree in classifying real-life mobile phone data. As a Random Forest is a combination of many decision trees, overcoming the problem of the overfitting of one decision tree, this one is chosen to use as non-linear classification model. A last classifier that is useful to compare with is Support Vector Machine, which is one of the most efficient algorithms with a vast variety of usage (Karamizadeh, Abdullah, Halimi, Shayan, & Rajabi, 2014). No previous assumption about the data is needed and this method can work very well when choosing the right kernel function.

3. Experimental Setup

This section firstly describes the data that is used for the research into the prediction for the next app category. After that, the programming parts of the research are described, starting with the needed preprocessing steps to get the right input for the models that are described thereafter. At the end, the evaluation methods are given.

3.1.Data

3.1.1. Description of the datasets

Two available datasets are used in this project. The data is gathered by an app that logged usage data. 124 participants were involved in the research between 21 February and 26 March 2019.

- phone_use_data.csv

This dataset consists of 586,792 rows of data, giving insights in the phone usage of 124 students. It contains information about the applications they opened: the names of the apps, the opening and ending times, whether a notification was received for these apps or not, and the session numbers. Also, the battery level of the phone and the user id are included in this dataset.

- app_categories.csv

This is an extension of phone use dataset, and clusters different applications into 59 categories. 1748 different applications are assigned to a category.

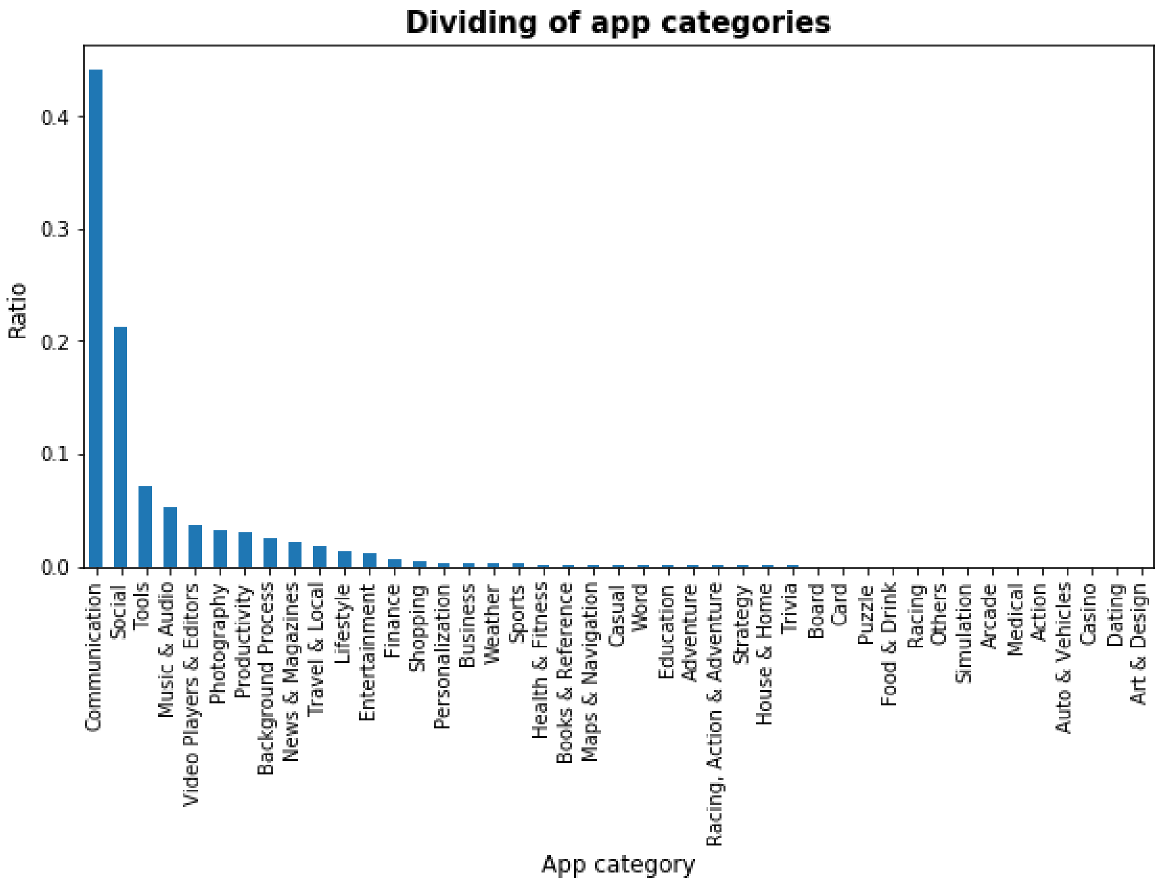
3.1.2. Preprocessing

All code for this research is written in the powerful programming language Python (Rossum, van, 1995) and can be found in Appendix V. The pandas package is used to combine the phone usage dataset with the app categories dataset (McKinney, 2010). Both datasets are merged on '*application'*, in such a way that the phone usage dataset is the same but a column is added containing the app category of the application. Not all apps were present in both datasets, so the combined data frame had less categories than existed in the dataset of app categories, and some categories were *NaN* values, because not all applications got a category assigned. Looking into the data, it seemed that there are two 'apps' that are not important in this research; *'com.ethica.logger'* (an app to get the data for this research) and *'com.android.systemui'* (no application), so all rows containing these are removed. The other *NaN* values were for apps that were not often used. These will later get the category *Low Frequency App*, but only after doing a few other steps. Furthermore, all duplicate rows are deleted.

As next step, some additional columns are added, using data of the existing columns. The day and hour of the column *startTime* are copied into new columns, named *Day* and *startHour*. *Duration* is made by subtracting *startTimeMillis* from *endTimeMillis*, which contain the start and end time of opening an app respectively, conversed into milliseconds. Columns of the previous app opened and the duration of the previous app are made as well, containing information about the application a user opened before the current one. As the aim of these variables is to see if they influence the current app category chosen, it is of no importance when the last app was opened at the previous day. Therefore, these features are made by not only looking at the different users, but also at the different days, so the first app opened by a user at a day gets the name *First_app*.

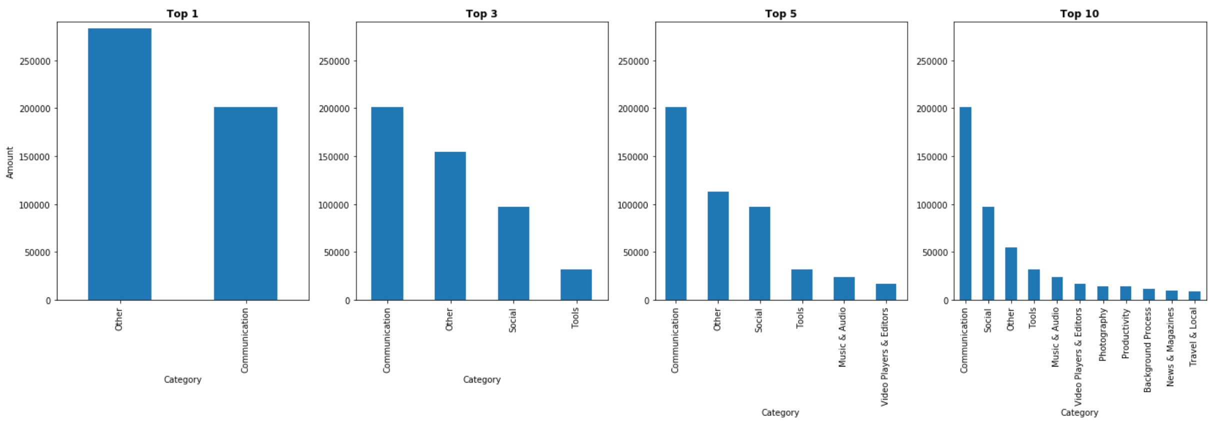
To see how the different categories are divided within the dataset, a bar plot is made of the normalized value counts (Figure 1). All plots in this thesis are made by using matplotlib (Barret, Hunter, Miller, Hsu, & Greenfield, 2005). An explanation of the content of every app category is in Table 5 (Appendix I).

Figure 1. *Bar plot of all different categories in the dataset*



In Figure 1, it is obvious that the data has a highly-skewed distribution, which is usually hard to predict. Most classifiers are designed to maximize the accuracy, so the majority class will be predicted much more often than the minority class (McCarthy, Zabar, & Weiss, 2005). Therefore, it is interesting to make not only a model for predicting one category out of all categories, but to focus on less categories to do that more precisely. For example, to predict *Communication* versus all other categories called *Other* will probably give a better result than predicting *Communication* versus the 43 other different categories. This is mentioned the top 1 category, and other predictions are made for the top 3, top 5, and top 10 as well. The column *category* in the dataset is used to make the needed additional columns by keeping only the top *k* categories and put all other in one and the same category *Other*. The dividing and content of the new columns is presented in the plots of Figure 2. The most important thing here is to see that when more categories are predicted, there are more in the 'long tail'. Not all categories can be read in this figure, but these categories can also be seen in the same order in Table 5 (Appendix I).

Figure 2. *Bar plots of new division of categories into top 1, top 3, top 5 and top 10 of categories respectively*



The last preprocessing steps can be done, starting with changing the *NaN* values of the column *all_categories* into *'Low Frequency App'*. It was not possible to do this earlier, as then *Low Frequency App* would be a new category to be predicted. As some techniques only accept numbers as inputs instead of strings, the boolean values of *notification* are changed into integers and dummies are made for *previous_app_opened*. For this research, only the columns *startHour*, *notification*, *previous_app_opened*, *previous_app_duration*, and all columns containing *category* are important. To avoid unnecessary repeating, the dataframe that is left is saved into a new csv file, which can be loaded easily.

Table 1 gives an overview of the attributes that are used in the models, with their possible values.

Table 1. *Overview of features used*

Variable name	***Name in dataframe***	***Explanation***	***Possible values***
$a_{hour} = a_0$	startHour	Hour when the app is opened	[0-23], integer
$a_{notification} = a_1$	notification	Whether a notification was sent for the application or not	{1, 0}; 1 if true, 0 if false

$a_{prev\_duration} = a_2$	previous_app_duration	Duration that previous app was opened, in milliseconds	[0, ∞], integer
$a_{prev\_app} = a_3$	previous_app_opened	The app name of the previous app opened	{app_1, app_2, …, app_M}, but changed to dummies: 1317 columns with {1,0}

The contextual vector x_i is a concatenation of all attribute values in a single vector for an opened application *i*:

$$x_i = [a_{hour,i}, a_{notification,i}, a_{prev\_app,i}, a_{prev\_duration,i}] \tag{1}$$

and $X = \{x_1, x_2, \ldots, x_N\}$. Given the vector x_i, the goal is to construct a model that predicts the target class label y, which is one of the app categories: $y = \{category_1,\ category_2,\ \ldots,\ category_M\}$.

The new data frame is loaded into Numpy Arrays (Oliphant, 2006) as inputs for train and test sets. 67 percent of the data is used in the train set and 33 percent in the test set, splitted randomly by using train_test_split of the package sklearn_model_selection (scikit-learn, sd).

3.2. Models

As explained in the related work section, the following models are made and compared with each other:

Table 2. *Overview of models made*

Model	*Discriminative*	*Generative*
Linear	Logistic Regression	Naïve Bayes
Non-linear	Random Forest, Support Vector Machine	

A description of every model will be given, theoretical and in formulas, and also the way it is programmed in Python with their (hyper)parameters. For this, scikit-learn is used, which is 'a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems' (Pedregosa, et al., 2011). The results are in the next chapter.

3.2.1. Logistic Regression (LR)

The roots of logistic regression are in the early 19th century (Cramer, 2002), but McFadden was the first who linked the multinomial logit to it in 1973 (McFadden, 2001). A single multi-class logistic regression model is based on a log-linear relationship between inputs and labels (Memisevic, Zach, Hinton, & Pollefeys, 2010) to be able to discriminate between all N classes (Leeuwen van & Brümmer, 2006). It finds a classification rule from the training data, such that a class label can be assigned to a new input x. Its scores are probabilistic outcomes based on a maximum likelihood argument (Karsmakers, Pelckmans, & Suykens, 2007), using the softmax function.

The conditional class probabilities are estimated via logit stochastic models, and is a combination of the softmax function and a regression model:

$$P(y \mid x_i) = \frac{1}{1+e^{-(\beta_0+\sum_{j=0}^{j=3}\beta_j a_{j,i})}} \tag{2}$$

In Equation 2, β_0 is the intercept and β_j is the coefficient associated with the explanatory variable $a_{j,i}$.

The package sklearn.linear_model.LogisticRegression is used in Python (scikit-learn, sd). As it is a multiclass problem, the *multi_class* parameter is set to *'multinomial'*, which uses the cross-entropy loss instead of the one-vs-rest scheme. Only the following *solver*s support the multinomial LR model: *'lbfgs'*, *'sag'*, *'newton-cg'*, and *'saga'*. The first three support only L2 regularization as *penalty*, and the last one can handle L1 and Elastic-Net as well. The Limited-memory BFGS (*'lbfgs'*) is a limited memory quasi-Newton method for large scale optimization (Liu & Nocedal, 1989) and is the default parameter for LR because of its robustness, so this one is chosen to implement. The SAGA solver is a promising optimization method according to Defazio, Bach and Lacoste-Julien (2014) as it has better theoretical convergence rates than other solvers, so this one is chosen as well, with different *penalty* functions. The maximum number of iterations is set to 200 for the *lbfgs* solver, but *saga* could only have 20 due to a memory error. To be able to have the same values every time the algorithm is run, a *random_state* is added of 42.

3.2.2. Naïve Bayes (NB)

The Naïve Bayes algorithm is often used in other studies that are related to the prediction of the next app someone is going to open (Shin, Hong, & Dey, 2012; Baeza-Yates, Jiang, Silvestri, & Harrison, 2015; Sarker, 2019), but also in other classification problems because of its simplicity and impressive classification accuracy (Farid, Zhang, Rahman, Hossain, & Strachan, 2014). It is a simple probability-based classifier based on Bayes' theorem that is around since the second half of the 18th century, with a strong assumption that the features are conditionally independent given the class variable (Zhang, 2004), so the effect of an attribute on a given class is independent of those of other attributes (Farid, Zhang, Rahman, Hossain, & Strachan, 2014). However, Zhang (2004) investigated naïve Bayes and proved that this method is still optimal, even if there exist some dependencies among attributes. NB calculates the probability that each of the features of a datapoint exists in each of the target classes and selects the category for which the probabilities are maximal by multiplying the posterior probabilities computed by using each feature. The class y_k for which $P(y_k|x_i)$ is maximized is called the Maximum Posteriori Hypothesis:

$$P(y_k|x_i) = \frac{P(x_i|y_k)P(y_k)}{P(x_i)} \qquad (3)$$

and the class will be chosen that fulfills

$$arg\ max\ P(y_k|x_i) \qquad (4)$$

In Python, there are four packages that can be used to apply Naïve Bayes, with their own assumptions of the dataset: Bernoulli, which assumes that all feature values are binary; Multinomial, which assumes to have discrete data; Complement, which corrects the ‘severe assumptions’ made by the standard Multinomial and is particularly suited for imbalanced datasets; and Gaussian, which assumes a normal distribution, and thus continuous data (scikit-learn, sd). As the app categories are represented in terms of their occurrences, both the Multinomial NB and Complement NB are made, with their corresponding packages of scikit-learn. The default parameters are used: the smoothing parameter *alpha* is 1.0; *fit_prior* is True, which means that the class prior probabilities are learned; and *class_prior* is None, meaning that the set prior probabilities of the classes are not set by hand. The Complement NB has an additional parameter *norm* to specify whether or not a second normalization of the weights is performed. Both options of this last parameter are tried in Python to see whether this has some effect on the accuracy.

3.2.3. Random Forest (RF)

A Random Forest is a classifier consisting of a combination of many decision trees where each tree casts a unit vote for the most popular class (Breiman, 2001). A Decision Tree is a classifier in the form of a tree structure, in which each node is either a leaf node or a decision node (Liu & Salvendy, 2007). Each leaf node has a class label, and all decision nodes have splits, testing the values of some functions of the data attributes. Selecting one of these splits is one of the main decisions to be made, which can be done by using a goodness measure that indicates the split’s classification power, such as Gini Gain, Information Gain, and Gain Ratio (Elzen, van den & Wijk, van, 2011; Liu & Salvendy, 2007; Liaw & Wiener, 2002).

Whereas in standard trees all nodes are split by using the best split among all variables, in a Random Forest each node is split using the best among a subset of predictors randomly chosen at that node (Liaw & Wiener, 2002). Decision Trees have been found to overfit training data, and RF overcomes this problem in many cases by selecting randomly a subset of features in each decision tree. RFs are especially good in learning non-linear relationships in high-dimensional class training data and can train rapidly (Chen, Ellis, & Velastin, 2011).

The package sklearn.ensemble.RandomForestClassifier is used in Python (scikit-learn, sd). The parameter *n_estimators* defines the number of trees in the forest. The default value is 10, so this one is used, as well as a larger value until a memory error is raised. The *criterion* parameter is the function to measure the quality of a split, and both options, Gini impurity (*'gini'*) and information gain (*'entropy'*), are tried. The default values of all other variables are retained, as these parameters are especially added to avoid overfitting, for example to define the maximum depth of the tree, or the minimum number of samples per split. However, as this research does not have a lot of features, these parameters can use their standard values.

3.2.4. Support Vector Machine (SVM)

Support Vector Machines are based on the statistical learning theory (Vapnik, 1998). A Support Vector Machine uses a kernel function to map input vectors into one feature space, possible with a higher dimension (Cai, Liu, Xu, & Chou, 2002). Within this feature space, an optimized linear decision boundary is made, called a hyperplane (Verplancke, et al., 2008), which separates the data into classes. SVM is a well-developed technique for binary classification, so multiclass SVMs are usually implemented by combining several of these binary SVMs (Duan & Keerthi, 2005), but extensions in the algorithm can handle multiclassification as well using additional parameters and constraints (Crammer & Singer, 2001). Scikit-learn has a package that has these additional parameters (scikit-learn, sd). The multi-class SVM formulated by Crammer and Singer can be implemented by using LinearCSV with *multi_class* = *'crammer_singer'*. Unfortunately, no result was possible due to a running time error. So, the SVC method is used, and this has lots of parameters. Choosing an appropriate *kernel* type is important as this can lead to a substantial difference in the capability of classifying (Belousov, Verzakov, & Frese, von, 2002; Barla, Odone, & Verri, 2003; Shunjie, Qubo, & Meng, 2012), and can be linear, polynomial, sigmoid, a radial basis function (rbf) or a self-made function. For this research, the default rbf is used, but sigmoid is tried as well. *gamma* defines the range of influence of a single training example in the feature space, and is an important parameter when the rbf kernel is used. The larger this value is, the closer other examples must be affected (scikit-learn). The default value '*auto*' is used for this, which is *1/n_features*, but *'scale'* is implemented as well, which uses *n_features*X.var()* as value. The *decision_function_shape* allows to transform the results of the one-against-all classifiers to one-vs-rest. Both are implemented in the models in which *gamma* = '*scale'*. The default values are used for the other parameters. An example is *class_weight*, which can give some classes more importance than others. This is not necessary in this research, because all categories have to be well predicted.

Before using SVM, two additional preprocessing steps had to be done. Firstly, the data is normalized, as different lengths in feature vectors could influence the outcome a lot (Herbrich & Graepel, 2001). This is done with the package *MinMaxScaler* (scikit-learn, sd), that scales all datapoints into values between the range (-1,1). Secondly, less data could be used to train and fit the model. The reason is that the implementation of sklearn is based on LIBSVM (Chang & Lin, 2011), and the fit time complexity is more than quadratic with the number of samples. Therefore, only 10,000 rows of data are used to train the model and 3,000 for testing. This causes a difference in the data compared to the data used for the other models. For example, there are less categories in the dataset. However, this does not have to be a problem, as SVM shows an optimal generalization ability (Belousov, Verzakov, & Frese, von, 2002).

3.3. Evaluation methods

The performances of the classifiers are optimized through the maximization of the accuracy, which is calculated by:

$$accuracy = \frac{\sum_{i=1}^{N} I(predicted\_class_i = real\_class_i)}{N} \quad (5)$$

where *I* is the indicator function, which returns 1 if the classes match and 0 otherwise.

The algorithms will not only be compared with the accuracy values of each other, but also with the majority class baseline, to see if they actually learnt something. This baseline is calculated by using equation (6), but with *predicted_class*$_i$ as the most occurring class in the dataset. So for example, the category *Communication* occurs most often in the *top 3* of the categories, which is 41.30 percent, so the baseline accuracy for the *top 3 categories* is 0.4130. As mentioned in the SVM section, this model uses less data and thus could have another baseline as well due to another number of the majority class. However, the same baseline is used as for the other models, as a baseline is something to compare all models with.

Different models are made to predict the top *k* categories, with *k* = {1, 3, 5, 10, all}. However, this is only done to see if there is any difference in the performance, not to choose another model for every *k*. Therefore, for every model, the average of these five predictions is calculated, and the best model is the one with the highest averaged accuracy.

To see whether there is a difference in the ability of predicting every separate category, the classifications of each class are individually analyzed, by calculating the precision and recall per class. This is only done for the best performing algorithm with the highest accuracy. Precision is the proportion of positive predicted cases that are actually correct, and recall is the proportion of real positive cases that were correctly predicted as positive (Powers, 2011). The formulas are the following:

$$precision = \frac{True\ Positive}{Total\ Predicted\ Positive} \quad (6)$$

$$recall = \frac{True\ Positive}{Total\ Actual\ Positive} \quad (7)$$

As the aim of this research is to help companies decide when they can send an advertisement to the phone screen for their mobile phone application, the cost of False Negative is higher than that of False Positive. In other words, it is better to predict *Communication* while this is not true than to predict something else while it is *Communication*. Therefore, recall is more important than precision. In Python, the package *metrics* is used to make a classification report (scikit-learn, sd). These tables are imported in Excel and the color scale green-yellow-red is used to see immediately which value is high and which is low.

Recall and precision only communicate a single aspect of the performance of a model (Ren, Amershi, Lee, Suh, & Williams, 2017). Therefore, confusion matrices are made as well for the best performing algorithm; these contain more details by contrasting the model predictions against the ground truth labels in a table of aggregated values (Ren, Amershi, Lee, Suh, & Williams, 2017). The confusion matrices are colored; the darker the color, the more instances are in that field. The True Positive values are all in the diagonals of these matrices, so these must actually be the darkest. The function of a confusion matrix made by scikit-learn is used and adjusted to make clear confusion matrices (scikit-learn, sd).

4. Results

In this section, the classification performance of the different models described in the previous section will be presented. First, the performances of LR, NB and RF with their different tested parameters are given. Then, a summary table is stated with one best performing model of LR, NB and RF, to see which technique works best in predicting the next app category with only the given features. The outcomes of the different combinations of features is stated afterwards. Finally, the classes of the best performing technique are further analyzed in confusion matrices and calculations of the recall and precision per class.

4.1. Performances of the different models

The tables that contain the different parameters tried for different methods are in Appendix II.

Table 6 (Appendix II) shows that Logistic Regression has an accuracy of 0.4478 for the best performing model and 0.4465 for the worst, averaged over all different predicted amounts of categories. Lots of the LR models were only able to predict the most occurring class, which leads to a score as low as the baseline, and some models predict even worse. This means that the input features were not linearly related to the log odds of the categories (Tu, 1996). The result contradicts the expectations, as LR is one of the most common approaches to solve a classification task (Memisevic, Zach, Hinton, & Pollefeys, 2010). Although the difference in performance of the parameters is only around 0.3 percent, it can be seen that the *saga* solver is slightly better than the *lbfgs* solver, which is in accordance with the research of Defazio, Bach and Lacoste-Julien (2014).

The accuracies of the models tried for Naïve Bayes are in Table 7 (Appendix II). It is clearly visible that the Complement Naïve Bayes model without second normalization outperforms the Multinomial model (0.4588 and 0.3587 averaged accuracy respectively). This is in accordance with the aims of Complement NB being the corrector of the Multinomial NB in some ways and to be better able to deal with imbalanced datasets (scikit-learn, sd). Another notice is that Complement NB performs better than the baseline in all cases, except in the top 1 categories. This is explainable because predicting the category *Communication* versus *Other* is actually a binary classification problem, while only multiclass NB models are made as the focus in this research is to have a model that can predict all categories well.

A remarkable division in the accuracies of the Random Forest Models is between the predictions of a lower amount of categories and the predictions of a larger amount (Table 8, Appendix II). The fewer categories that need to be predicted in this problem, the better an RF model works according to these results. This is in accordance with the known difficulty in predicting highly imbalanced classes

(McCarthy, Zabar, & Weiss, 2005), but RF even has an accuracy below the baseline for predicting ten categories or more. This is intuitively possible, as the more classes that need to be predicted, the more misclassification errors are possible as well. There is no large difference in the performances of the different parameters that were tested; all models have an averaged accuracy between 0.4657 and 0.4703. A higher number of estimators does not have the expected improved accuracy, and information gain (‘*entropy*’) performs a little bit better than Gini impurity as measurement for the quality of split.

It is obviously visible in Table 9 (Appendix II) that there is a large difference in performance of the different models of SVMs; the averaged accuracies differ from 0.4438 to 0.5564. This is in accordance with the expectation that kernel functions influence the performance a lot (Belousov, Verzakov, & Frese, von, 2002; Barla, Odone, & Verri, 2003), as well as the selection of the parameters (Shunjie, Qubo, & Meng, 2012). Support Vector Machine models using the default kernel *‘rbf’* and the gamma function *‘scale’* perform better than models using another kernel or gamma function. There is no difference in the performance of a model that classifies one-versus-one or one-versus-rest, which is also as expected (scikit-learn, sd).

4.2. Comparison of methods

The best performing models of all four different classifiers that predicted the next phone application category are chosen to represent that classifier, and the corresponding accuracies are in Table 3. In this table, a red color means that the accuracy is lower than the baseline, and a green one is a larger accuracy.

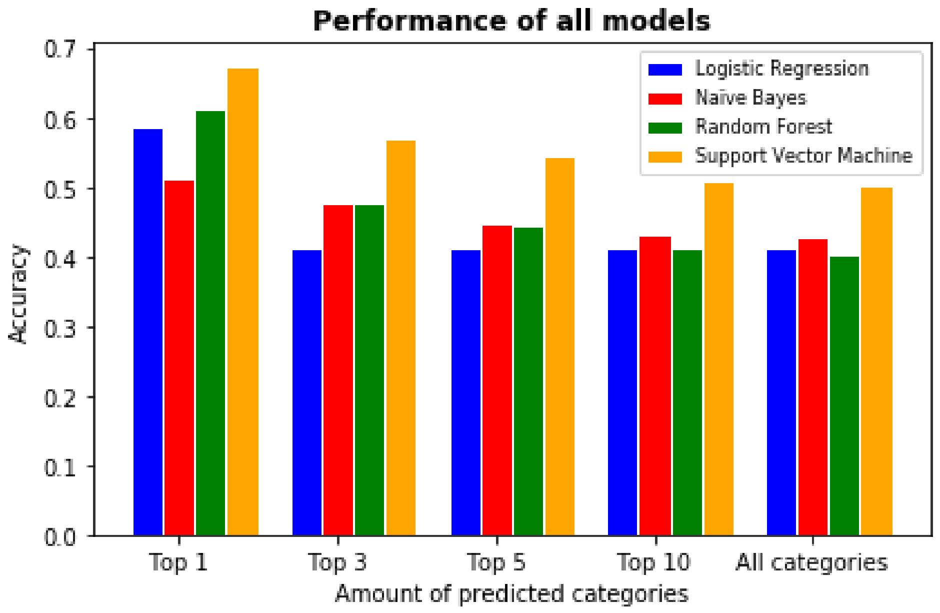
Table 3. *Accuracies of all classification models predicting the phone application category*

Classification Model	**Accuracies**				***All***	
	Top 1	***Top 3***	***Top 5***	***Top 10***	***categories***	***Mean***
Logistic Regression	0.5870	🟥 0.4120	🟥 0.4116	🟥 0.4109	🟥 0.4108	0.4465
Naïve Bayes	🟥 0.5112	🟩 0.4757	🟩 0.4485	🟩 0.4308	🟩 0.4280	0.4588
Random Forest	🟩 0.6133	🟩 0.4774	🟩 0.4459	🟥 0.4121	🟥 0.4028	0.4703
Support Vector Machine	🟩 **0.6737**	🟩 **0.5690**	🟩 **0.5440**	🟩 **0.5073**	🟩 **0.5030**	**0.5594**
Baseline	0.5870	0.4130	0.4130	0.4130	0.4130	0.4478

Looking at the results in Table 3, it is clear that Support Vector Machine outperforms the others and Logistic Regression performs badly. The model of Naïve Bayes has more accuracies above the baseline, but on average the model of Random Forest is better to use.

To see the differences more clearly, the outcomes are visualized in a bar plot (Figure 3).

Figure 3. *Bar plot of the accuracies of all classification models predicting the phone application category*



In Figure 3, it is visible that NB, RF and SVM perform worse the more categories these have to predict, whereas LR stays constant upward of top 3. The large difference of SVM compared to the others is clearly visible.

4.3. Input features

The performance of the Support Vector Machine with all possible selections of the four features hour, notification, previous application duration and previous app opened is stated in Table 10 (Appendix III). The accuracies did not improve when using less features, which was expected as four features are only a few. The most obvious thing in this table is that the duration of the previous app does not have any influence on the performance of the model; whether a_2 is added or not, the accuracy was the same. The best feature on its own is the previous app opened, but notification is very useful as well; notification and previous app opened lead together to almost the same accuracy as using all features. The hour of day gives the last additional approximately 0.5 percent above the accuracy.

4.4. Additional performance information per category

To see clearly the difference in performance of each category, the classification reports and normalized confusion matrices are made per top k categories that are predicted by the SVM model and placed in Appendix IV. In total, there are 44 categories (including *Low Frequency App*), so it was not possible to make a clear confusion matrix for *All categories*. Therefore, the figures of the top 10 are stated below instead of these of all categories, as well as descriptions of the most remarkable things in performance of the other predicted amounts of categories.

The classification reports show that *Communication* is best predicted in every top *k* categories, except in top 1. However, even in the top 1 classification report this recall is 0.64 (Figure 5, Appendix IV), which means that 64 percent of all *Communication* data is predicted as such. This recall only rises the more categories that are predicted, to 0.75 in all categories. However, the precision goes down from 0.59 to 0.52. This means that *Communication* is also often predicted while it was actually another category. This is clearly visible in the confusion matrices, where the box with the real and predicted category *Communication* does have the highest value, but the whole column of predicted *Communication* has large values as well. Fortunately, in every confusion matrix, most other True Positives do have a large value as well, which is visible through the darker color of the diagonal.

Another remarkable thing is that the more instances a category has, the better its recall value is, where the amount of instances are represented by the support values in the classification reports. However, there are some contradictions, the first being in the prediction of the top 5 categories, which shows that the category *Music & Audio* has a relatively large recall value (0.55), while the support value is almost the lowest (Table 13, Appendix IV).

Below is the classification report of the top 10 predicted categories (Table 4).

Table 4. *Classification Report SVM Top 10*

	Precision	*Recall*	*Support*
Background Process	0.29	0.03	74
Communication	0.53	0.75	1219
Music & Audio	0.64	0.55	164
News & Magazines	0.42	0.51	61
Other	0.53	0.33	339
Photography	0.53	0.26	100
Productivity	0.36	0.04	93
Social	0.44	0.43	565
Tools	0.5	0.29	219
Travel & Local	0.53	0.44	54
Video Players & Editors	0.33	0.14	112

Table 4 shows a large difference in the recall values; 0.03 for the lowest and 0.75 for the best class. Although *News & Magazines* and *Travel & Local* have even lower support values than the worst scoring *Background Process,* these have a recall value around fifty percent. To get more insight in which category is predicted for every actual category, the confusion matrix is visible in Figure 4.

Figure 4. *Confusion Matrix SVM Top 10*

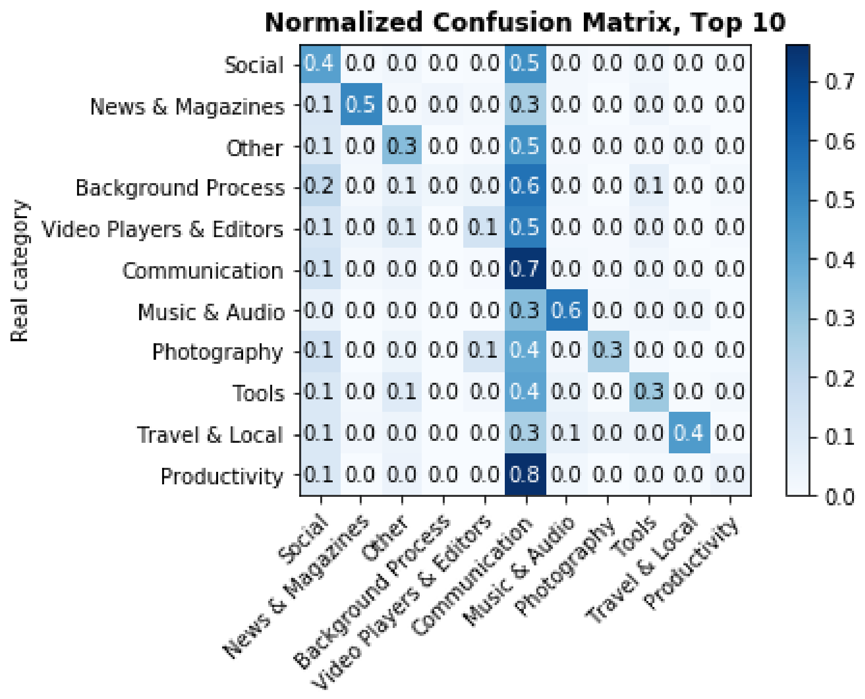


Figure 4 shows again that *Communication* is predicted for most instances, and only the categories *News & Magazines* and *Music & Audio* were able to predict their own category more often than *Communication*. The only other category that predicted *Communication* according to this rounded normalized matrix is *Social*, which is intuitively true as both categories are related to each other.

Looking at the classification report of all categories (Table 14, Appendix IV), it is notable that *Communication* still has a recall value of 0.75. So, although 34 more classes are possible to predict, this value is the same as when predicting only 10 categories (Table 4). Categories with low support also have a low recall and precision value (0 in most cases).

5. Discussion

The goal of this study was to investigate the extent to which the next mobile phone application category that someone will open could be predicted, while using only mobile phone data that is accessible for companies. So, to build a model that companies can use by deciding when they can send an advertisement for their app. This section gives the findings of the research and relates these to what was already known of this problem in the literature. It also mentions the limitations of this research and suggestions for further research.

5.1. Findings

The first question investigated which model could best predict the next phone application category. Based on literature, the following algorithms were chosen to be built: Logistic Regression, Naïve Bayes, Random Forest, and Support Vector Machine. Different parameters were tested for every model to see which worked best and the following input features were used: time (in hours), previous app opened, duration of previous app, and presence of notification. The best model was Support Vector Machine with the radial basis function kernel and '*scale'* for gamma parameter. It was not surprising that SVM outperformed the other three algorithms. Research to differences between SVM and LR concluded that SVM required less variables than LR to achieve the same accuracy (Verplancke, et al., 2008; Salazar, Vélez, & Salazar, 2012), thus, having only four input features, it was expected that SVM outperformed LR. Alsaleem (2011) investigated the difference in performance of SVM and NB in text categorization and found that SVM outperformed NB regards to F1, recall and precision measures. Although text categorization is not the same as mobile phone categorization, it is still a classification task and therefore comparable. However, when more features were used, NB would perform better according to Hassan, Rafi, and Shahid (2011). They found that NB gives better results than SVM when external enriching is used through any external knowledge base for text classification (Hassan, Rafi, & Shahid Shaikh, 2011). A study into the difference in performance of SVM and RF is done by Kremic and Subasi (2016). They presented both algorithms in facial recognition, which is also a classification task. Dependent on the choice for the kernel function for SVM, this one could outperform RF a bit (Kremic & Subasi, 2016), whereas another research found a consistent outperforming of SVM compared to RF in a classification task into vehicle type categorization (Chen, Ellis, & Velastin, 2011).

So, SVM was the best model to use for predicting the next phone app category. At the start of this research, a division was made between linear-nonlinear and discriminative-generative models. However, there is no clear preference for one. In this case, discriminative and non-linear worked best, but neither the other non-linear nor the other discriminative model worked well.

The second question investigated the usefulness of all four features for the prediction of the next phone application category. All combinations of the features were tested in the SVM model to see if the accuracy changed. The order of importance of the four features from most important to least is the following: previous app opened, notification, hour of day, duration of previous app. This last one didn't have had any influence at all. Although it was worth trying this feature, it can intuitively be understood that it didn't have a relation with the next app chosen. The large credit of the previous application opened by someone on the next one was expected, as it was already proven by other researches in this field (Shin, Hong, & Dey, 2012; Huang, Zhang, Ma, & Chen, 2012). Also, hour of day is proven to have an influence on the app that would be chosen (Böhmer, Hecht, Schöning, Krüger, & Bauer, 2011; Shin, Hong, & Dey, 2012), but it was expected to have more potency. An explanation for this can be the relatively short time period the dataset covers (34 different days). The last variable, notification, seemed to have a large influence on the next app category that was chosen. However, in the dataset used for this research, only ten percent of the opened apps received a notification for it, and these were mostly send by applications in the categories *Communication*, *Social*, *Tools*, and *Productivity*. Moreover, only the apps that were actually opened were visible; not the apps that sent a notification but were ignored by the user. According to Mehrotra, Hendley and Musolesi (2016), users don't accept all notifications they get and even dismiss those that are not useful or relevant for them. So, the presence of a notification has an influence on the accuracy of the model, but this doesn't say anything about the influence of receiving a notification. Summarized, the previous app opened and hour of day influence the choice of the next app category, and so does notification in this specific research but this can be different when using other data.

The last part of this research focused on the difference in the predictability of the different categories. This is done by making classification reports containing the precision, recall and support values per category. In this research, it is important to get all instances that can possibly have a specific category, and assigning this category while another was the actual one is not a big problem; it is acceptable to a certain extent to send an advertisement for an application while the user planned to open another category. Therefore, the recall value was more important than the precision. The best recall value belonged to the most occurring category in the dataset; *Communication*, followed by *Music & Audio*, *News & Magazines, Travel & Local*, and *Social*. Other categories had a low support value, causing the algorithm having difficulties with predicting these categories, so these recall values were low. To see which categories were mixed up with each other, confusion matrices were made as well, showing the predicted categories for every actual category. It was clear that *Communication* was predicted too often, and most categories were even more predicted as *Communication* than their actual category. One category that was mixed up with *Communication* is *Social*. The high number of misclassifications between these is argumentative, as these are related to each other, and can thus have features that exhibit significant similarity.

Combining the fact of having 44 different application categories in the dataset with the aim to find the extent to which the next phone application category could be predicted, a division was made in all questions between five different prediction problems: top 1, top 3, top 5, top 10 and all categories. For example, the category *Communication* occurred most often in the dataset, so for top 1, this category was predicted versus all other categories. It turned out that the more categories that needed to be predicted, the less accurate the classification model was. Moreover, having a larger choice of categories that could be predicted, the model chose for the most occurring class *Communication* more often, which leaded to a high recall score for this category, but for a lower one for the others.

The best performing model had an accuracy of around 55 percent, which was around ten percent above the baseline. Although this score is low, the research still has added value. This is to our knowledge the first research into a prediction model for the next phone application category that companies can use, so further research can build on this and try to improve the accuracy. Moreover, the aim of this research was to make a model that predicted as much categories right as possible, but a company usually only has an application in one category. The model can easily be changed to focus on an accurate prediction for one of the categories, for example by adding weights of importance (scikit-learn, sd). A research of McCarthy, Zabar and Weiss (2005) can be used for this as well, as they built models that focus on minority classes in highly-skewed class distributions instead of maximizing accuracy, by comparing cost-sensitive learning with up-sampling and down-sampling.

5.2. Drawbacks and suggestions on further research

A first limitation of this research was in the data that is used. The dataset about the mobile phone usage behavior contained only information about students and was gathered in a relatively short period. For the purpose of generalization, it would be better to have data of more different people gathered over a longer period of time. Besides, the dataset that contained the application categories didn’t assign a category to all apps, making it difficult for owners of unknown apps to use this model. Therefore, a suggestion for additional research would be to build a model that can automatically assign a category to all apps.

Another drawback was the lack of memory in Python and time, making it impossible in the first place to use all data for the SVM model. Therefore, not all categories of the data were used for this model, and some had a support of only one (Table 14, Appendix IV). This could have affected the performance, and it would be better if all data could be used for this model as well. Secondary, grid search was not applicable, while Shunjie, Qubo and Meng (2012) recommended double grid search on choosing the optimal parameters of the rbf kernel. Although the choice of parameters was based on literature, it would be better to try all possibilities and compare the outcomes with the expectations. A suggestion for further research would be to try the other parameters in the SVM model to see if any improvement can be made. Moreover, other methods can be used as well, or combinations of methods, which is also done in other research (Baeza-Yates, Jiang, Silvestri, & Harrison, 2015; Sarker, 2019). Although a neural network is very similar to SVM (Kremic & Subasi, 2016), it is still worth trying as it has the ability to implicitly detect complex nonlinear relationships and has multiple training algorithms (Tu, 1996).

A last important thing to mention is that this research is built on the expectation that companies can have access to all four features used, but there was no research done that checked this. Therefore, a last suggestion for further research would be to get insights into the mobile phone usage data that app owners have access to. They may even have access to other data that was not available in this research but could have improved the model. So, new research can focus on filling this gap: knowing which data companies can use and which of these features can help in the prediction of the choice of the next app category.

6. Conclusion

As smartphones are used in almost every aspect of people's lives, more and more mobile phone applications are developed to support this. Due to this expanding choice of apps, research is done to simplify the choice for the user by making automated recommendations based on predictions of the next application that someone will use. However, to help app owners in their competition, it was interesting to make a prediction model for them as well, which they can use to decide when they can send an advertisement banner to the phone screen of the user. Instead of focusing on the next app someone will use, the focus was on the next app category, and only data is used in the model that companies presumably have access to. To investigate the possibility of building such a model, the research question of this thesis was as follows: ***To what extent can the phone application category that someone will open be predicted, while using only mobile phone data that is accessible for companies?*** The following three sub questions helped by answering this main question:

1. *Which model predicts the next phone application best?*
2. *Which features are useful in the prediction of the next phone application category?*
3. *Is there any difference in the predictability of the different categories?*

Using a dataset containing mobile phone usage data with categories assigned to the apps, four classification models are compared: Logistic Regression, Naïve Bayes, Random Forest and Support Vector Machine. It turned out that SVM had the highest accuracy; around 55 percent of the data was predicted in the right category. The features used in this model, in order of importance, were the following: previous app opened, notification, hour of day, and duration of previous app. The influence of previous app opened and hour of day were already proven in other researches. The presence of a notification also influenced the prediction, but this is no proof that sending a notification influences the choice of the user a lot, as this dataset only contained data about apps that were already opened. Looking at the recall values, there was a large difference in the predictability of the different categories. The most occurring category in the dataset was best predicted and instances that didn't occur much also had a low recall value. The confusion matrices showed that the most occurring category was also often predicted for instances that didn't have that category. Also, the more categories that needed to be predicted, the lower the accuracy was.

Concluding, it is possible to predict the phone application category that someone will open, while using only mobile phone data that is accessible for companies. However, it is dependent on which category an app belongs to if this specific model can work or not. It is recommended for app owners to change the model a bit by giving their own app category more weight of importance, so that the model tries to achieve a high recall value for this specific category.

References

Alsaleem, S. (2011). Automated Arabic Text Categorization Using SVM and NB. *International Arab Journal of e-Technology, 2*(2), 124-128.

Baeza-Yates, R., Jiang, D., Silvestri, F., & Harrison, B. (2015). Predicting the next app that you are going to use. *Proceeding WSDM '15,* 285-294. doi:10.1145/2684822.2685302

Barla, A., Odone, F., & Verri, A. (2003). Histogram intersection kernel for image classification. *Proceedings 2003 International Conference on Image Processing*, 513-516. doi:10.1109/ICIP.2003.1247294

Barret, P., Hunter, J., Miller, J., Hsu, J.-C., & Greenfield, P. (2005). Matplotlib - A Portable Python Plotting Package. *Astronomical Data Analysis Software and Systems XIV, 347*. Retrieved from https://matplotlib.org/

Becker, B., Kohavi, R., & Sommerfield, D. (2001). Visualizing the simple bayesian classifier. In U. Fayyad, G. Grinstein, & A. Wierse, *Information Visualization in Data Mining and Knowledge Discovery* (pp. 237-249).

Belousov, A., Verzakov, S., & Frese, von, J. (2002). A flexible classification approach with optimal generalisation performance: support vector machines. *Chemometrics and Intelligent Laboratory Systems, 64*(1), 15-25. doi:10.1016/S0169-7439(02)00046-1

Böhmer, M., Hecht, B., Schöning, J., Krüger, A., & Bauer, G. (2011). Falling Asleep with Angry Birds, Facebook and Kindle - A Large Scale Study on Mobile Application Usage. *Proceeding MobileHCI '11,* 47-56. doi:10.1145/2037373.2037383

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32. doi:10.1023/A:1010933404324

Cai, Y.-D., Liu, X.-J., Xu, X.-b., & Chou, K.-C. (2002). Prediction of protein structural classes by support vector machines. *Computers and Chemistry, 26,* 293-296.

Cao, H., & Lin, M. (2017, June). Mining smartphone data for app usage prediction and recommendations: A survey. *Pervasive and Mobile Computing*(37), 1-22. doi:10.1016/j.pmcj.2017.01.007

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*(3), Article 27. doi:10.1145/1961189.1961199

Chen, Z., Ellis, T., & Velastin, S. (2011). Vehicle Type Categorization: A comparison of classification schemes. *14th International IEEE Conference on Intelligent Transportation Systems*, 74-79. doi:10.1109/ITSC.2011.6083075

Cramer, J. (2002). The Origins of Logistic Regression. *Tinbergen Institute Working Paper*.

Crammer, K., & Singer, Y. (2001). On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research, 2,* 265-292.

Defazio, A., Bach, F., & Lacoste-Julien, S. (2014). SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. *Advances in Neural Information Processing Systems, 27,* 1646-1654.

Do, T., & Gatica-Perez, D. (2014, June). Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*(12), 79-91. doi:10.1016/j.pmcj.2013.03.006

Duan, K.-B., & Keerthi, S. (2005). Which Is the Best Multiclass SVM Method? An Empirical Study. *International Workshop on Multiple Classifier Systems*, 278-285. doi:10.1007/11494683_28

Elzen, van den, S., & Wijk, van, J. (2011). BaobabView: Interactive Construction and Analysis of Decision Trees. *IEEE Conference on Visual Analytics Science and Technology*, 151-160.

Farid, D., Zhang, L., Rahman, C., Hossain, M., & Strachan, R. (2014). Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. *Expert Systems with Applications, 41*, 1937-1946. doi:10.1016/j.eswa.2013.08.089

Hassan, S., Rafi, M., & Shahid Shaikh, M. (2011). Comparing SVM and Naïve Bayes Classifiers for Text Categorization with Wikitology as knowledge enrichment. *IEEE 14th International Multitopic Conference*, 31-34. doi:10.1109/INMIC.2011.6151495

Herbrich, R., & Graepel, T. (2001). A PAC-Bayesian Margin Bound for Linear Classifiers: Why SVMs work. *Advances in Neural Information System Processing, 13*.

Huang, K., Zhang, C., Ma, X., & Chen, G. (2012). Predicting mobile application usage using contextual information. *Proceeding UbiComp '12*, 1059-1065. doi:10.1145/2370216.2370442

Kamisaka, D., Muramatsu, S., Yokoyama, H., & Iwamoto, T. (2009). Operation Prediction for Context-Aware User Interfaces of Mobile Phones. *Ninth Annual International Symposium on Applications and the Internet*, 16-22. doi:10.1109/SAINT.2009.12

Karamizadeh, S., Abdullah, S., Halimi, M., Shayan, J., & Rajabi, M. (2014). Advantage and Drawback of Support Vector Machine Functionality. *International Conference on Computer, Communication, and Control Technology*, 63-65. doi:10.1109/I4CT.2014.6914146

Karsmakers, P., Pelckmans, K., & Suykens, J. (2007). Multi-class kernel logistic regression: a fixed-size implementation. *Proceedings of International Joint Conference on Neural Networks*, 1756-1761. doi:10.1109/IJCNN.2007.4371223

Kremic, E., & Subasi, A. (2016). Performance of Random Forest and SVM in Face Recognition. *The International Arab Journal of Information Technology, 13*(2), 287-293.

Leeuwen van, D. A., & Brümmer, N. (2006). Channel-dependent GMM and Multi-class Logistic Regression models for language recognition. *IEEE Odyssey - The Speaker and Language Recognition Workshop*. doi:10.1109/ODYSSEY.2006.248094

Liao, Z.-X., Pan, Y.-C., Peng, W.-C., & Lei, P.-R. (2013). On Mining Mobile Apps Usage Behavior for Predicting Apps Usage in Smartphones. *Proceeding CIKM '13*, 609-618. doi:10.1145/2505515.2505529

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R news, 2*, 18-22.

Lim, K.-W., Secci, S., Tabourier, L., & Tebbani, B. (2016, December). Characterizing and predicting mobile application usage. *Computer Communications*(95), 82-94. doi:10.1016/j.comcom.2016.04.026

Liu, D., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming, 45*(1-3), 503-528.

Liu, Y., & Salvendy, G. (2007). Interactive visual decision tree classification. *Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques*, 92-105.

McCarthy, K., Zabar, B., & Weiss, G. (2005). Does Cost-Sensitive Learning Beat Sampling for Classifying Rare Classes? *Proceedings of the 1st international workshop on Utility-based data mining*, 69-77. doi:10.1145/1089827.1089836

McFadden, D. (2001). Economic Choices. *American Economic Review, 91*(3), 352-370.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 51-56.

Mehrotra, A., Hendley, R., & Musolesi, M. (2016). PrefMiner: Mining User's Preferences for Intelligent Mobile Notification Management. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 1223-1234. doi:10.1145/2971648.2971747

Memisevic, R., Zach, C., Hinton, G., & Pollefeys, M. (2010). Gated Softmax Classification. *Advances in Neural Information Processing Systems*, *23*, 1603-1611.

Ng, A., & Jordan, M. (2002). On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in neural information processing systems, 14*, 841-848.

Oliphant, T. (2006). *Guide to NumPy.* Trelgol Publishing. Retrieved from http://numpy.scipy.org/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Powers, D. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63.

Ren, D., Amershi, S., Lee, B., Suh, J., & Williams, J. (2017). Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers. *IEEE Transactions on visualization and computer graphics*, *23*(1), 61-70. doi:10.1109/TVCG.2016.2598828

Rossum, van, G. (1995). *Python tutorial.* Technical Report.

Salazar, D., Vélez, J., & Salazar, J. (2012). Comparison between SVM and Logistic Regression: Which One is Better to Discriminate? *Revista Colombiana de Estadística, 35*(2), 223-237.

Sarker, I. (2019). A machine learning based robust prediction model for real-life mobile phone data. *Internet of Things, 5*, 180-193. doi:10.1016/j.iot.2019.01.007

scikit-learn. (n.d.). *scikit-learn.org*. Retrieved from https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

Shin, C., Hong, J.-H., & Dey, A. (2012). Understanding and prediction of mobile application usage for smart phones. *Proceeding UbiComp '12*, 173-182. doi:10.1145/2370216.2370243

Shunjie, H., Qubo, C., & Meng, H. (2012). Parameter selection in SVM with RBF kernel function. *World Automation Congress 2012*.

Smit, E., Noort, G. v., & Voorveld, H. (2014). Understanding online behavioural advertising: User knowledge, privacy concerns and online coping behaviour in Europe. *Computers in Human Behavior, 32*, 15-22. doi:10.1016/j.chb.2013.11.008

Srihari, S. (2018, September). Machine Learning: Generative and Discriminative Models. Buffalo. Retrieved November 9, 2019, from https://cedar.buffalo.edu/~srihari/CSE574/Discriminative-Generative.pdf

Tan, C., Liu, Q., Chen, E., & Xiong, H. (2012). Prediction for Mobile Application Usage Patterns. *Nokia MDC Workshop*. Newcastle, UK.

Tu, J. V. (1996). Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes. *Journal of Clinical Epidemiology, 49*(11), 1225-1231. doi:10.1016/S0895-4356(96)00002-9

Vapnik, V. (1998). Statistical learning theory.

Verplancke, T., Van Looy, S., Benoit, D., Vansteelandt, S., Depuydt, P., De Turck, F., & Decruyenaere, J. (2008). Support vector machine versus logistic regression modeling for prediction of hospital mortalitiy in critically ill patients with haematological malignancies. *BMC Medical Informatics and Decision Making, 8*, 56-64. doi:10.1186/1472-6947-8-56

Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z., . . . Choudhury, T. (2013). Preference, Context and Communities: A Multi-faceted Approach to Predicting Smartphone App Usage Patterns. *Proceeding ISWC '13*, 69-76. doi:10.1145/2493988.2494333

Zhang, H. (2004). The Optimality of Naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*.

Zhu, H., Chen, E., Xiong, H., Cao, H., & Tian, J. (2014). Mobile App Classification with Enriched Contextual Information. *IEEE Transactions on Mobile Computing, 13*(7), 1550-1563.

Appendix I: Information of the phone application categories

In Table 5, all categories are given that were used as classes in the classification methods, in order of popularity in the dataset.

Table 5. *Order of all phone application categories including examples*

Category	*Application examples*
1. Communication	Messaging, WhatsApp
2. Social	Facebook, Snapchat, Instagram
3. Tools	Calculator, Clock, Settings
4. Music & Audio	Spotify
5. Video Players & Editors	YouTube, Video
6. Photography	Gallery3d, Camera
7. Productivity	Calendar, Notes, Memo
8. Background Process	Telecom, Package installer, Vending
9. News & Magazines	Twitter, Reddit, NOS
10. Travel & Local	Maps, NS
11. Lifestyle	MobileDNA, Tinder
12. Entertainment	Mediaclient, TVShowtime, Dumpert
13. Finance	ING, Rabomobiel, ABNAmro
14. Shopping	AliExpress, Zalando, Marktplaats
15. Personalization	Themestore, Parallel Space
16. Business	Companyportal, Slack
17. Weather	Buienalarm, Buienradar
18. Sports	Footballaddicts livescore, Skitracker
19. Health & Fitness	Bluelightfilter, Sleepcycle, Shealth
20. Books & Reference	Audible application, Books, Wikipedia
21. Maps & Navigation	DB Navigator, Flitsmeister
22. Casual	Candycrushsaga, BestFiends
23. Word	Wordfeud
24. Education	Duolingo, Quizletandroid, Blackboard
25. Adventure	PokemonGo
26. Racing, Action & Adventure	R3row
27. Strategy	DeMol, Clashofclans
28. House & Home	Peel smart remote
29. Trivia	Intermedia, Trivia crack
30. Board	Sudoku
31. Card	Heartstone
32. Puzzle	Tenten, Bubblewitch
33. Food & Drink	Deliveroo, Tgtg
34. Racing	Hillclimb
35. Others	Zeropage
36. Simulation	Choices
37. Arcade	Subwaysurf, Templerun
38. Medical	Pillreminder
39. Action	Tencent
40. Auto & Vehicles	Autoscout
41. Casino	MyTalkingAngela
42. Dating	Once
43. Art & Design	Sketch
44. Low Frequency App	Other apps

Appendix II: Results different parameters

All tables below contain the accuracies of the different models tried, with the baseline as bottom row. To be able to see clearly which model performs better than this baseline and which not, the values are given a color; red means that the accuracy is lower than the baseline, and green is a higher value. If no color is given, the accuracy is the same as the baseline. Every column is a different amount of classes to predict, with in bold its highest accuracy.

Table 6. *Accuracies of Logistic Regression*

Logistic Regression Model	*Top 1*	*Top 3*	*Top 5*	*Top 10*	*All*	*Mean*
Solver = 'lbfgs', max_iter = 100	0.5870	0.4130	0.4109	0.4086	0.4130	0.4465
Solver = 'lbfgs', max_iter = 200	0.5870	0.4130	0.4109	0.4087	0.4130	0.4465
Solver = 'saga', max_iter = 20	0.5870	0.4130	**0.4130**	**0.4130**	0.4130	0.4478
Solver = 'saga', max_iter = 10, penalty = L1	0.5870	0.4130	**0.4130**	**0.4130**	0.4130	0.4478
Baseline	0.5870	0.4130	0.4130	0.4130	0.4130	0.4478

Table 7. *Accuracies of Naïve Bayes*

Naïve Bayes Model	*Top 1*	*Top 3*	*Top 5*	*Top 10*	*All*	*Mean*
Multinomial	**0.5283**	0.4487	0.3962	0.2301	0.1901	0.3587
Complement, norm = False	0.5112	**0.4757**	**0.4485**	**0.4308**	**0.4280**	0.4588
Complement, norm = True	0.4592	0.4167	0.3940	0.3969	0.3970	0.4128
Baseline	0.5870	0.4130	0.4130	0.4130	0.4130	0.4478

Table 8. *Accuracies of Random Forest*

Random Forest Model	***Top 1***	***Top 3***	***Top 5***	***Top 10***	***All***	***Mean***
n_estimators = 10, criterion = 'gini'	0.6130	**0.4773**	0.4447	0.4119	**0.4037**	0.4701
n_estimators = 50, criterion = 'gini'	0.6148	0.4722	0.4391	0.4051	0.3975	0.4657
n_estimators = 10, criterion = 'entropy'	0.6133	0.4774	**0.4459**	**0.4121**	0.4028	0.4703
n_estimators = 50, criterion = 'entropy'	**0.6154**	0.4739	0.4420	0.4081	0.3990	0.4677
Baseline	0.5870	0.4130	0.4130	0.4130	0.4130	0.4478

Table 9. *Accuracies of Support Vector Machine*

Support Vector Machine Model	***Top 1***	***Top 3***	***Top 5***	***Top 10***	***All***	***Mean***
gamma = 'scale', decision_function_shape = 'ovo'	**0.6737**	**0.5690**	**0.5440**	**0.5073**	**0.5030**	0.5594
gamma = 'scale', decision_function_shape = 'ovr'	**0.6737**	**0.5690**	**0.5440**	**0.5073**	**0.5030**	0.5594
gamma = 'auto', decision_function_shape = 'ovo'	0.6313	0.5187	0.4063	0.4063	0.4063	0.4738
kernel = 'sigmoid', gamma = 'scale', decision_function_shape = 'ovo'	0.5937	0.4063	0.4063	0.4063	0.4063	0.4438
Baseline	0.5870	0.4130	0.4130	0.4130	0.4130	0.4478

Appendix III: Results different features tested

Table 10 shows the accuracies of the SVM model when using less input features. All combinations of the following four input features are used:

$a_0 = a_{hour}$ $a_1 = a_{notification}$ $a_2 = a_{prev\_duration}$ $a_3 = a_{prev\_app}$

To see immediately which combination works well and which not, the color scale green-yellow-red is used, green meaning a high value and red a low one.

Table 10. *Accuracies SVM with different inputs*

Input features	*Accuracies*				
	Top 1	*Top 3*	*Top 5*	*Top 10*	*All categories*
1. a0+a1+a2	0.6220	0.4083	0.4063	0.4063	0.4063
2. a0+a1+a3	0.6737	0.5690	0.5440	0.5073	0.5030
3. a0+a2+a3	0.6623	0.5510	0.5257	0.4910	0.4850
4. a1+a2+a3	0.6707	0.5647	0.5387	0.5040	0.4980
5. a0+a1	0.6220	0.4093	0.4063	0.4063	0.4063
6. a0+a2	0.5937	0.4063	0.4063	0.4063	0.4063
7. a0+a3	0.6623	0.5510	0.5257	0.4910	0.4850
8. a1+a2	0.6220	0.4063	0.4063	0.4063	0.4063
9. a1+a3	0.6707	0.5647	0.5387	0.5040	0.4980
10. a2+a3	0.6597	0.5487	0.5240	0.4903	0.4843
11. a0	0.5937	0.4090	0.4063	0.4063	0.4063
12. a1	0.6220	0.4063	0.4063	0.4063	0.4063
13. a2	0.5937	0.4046	0.4050	0.4060	0.4060
14. a3	0.6597	0.5487	0.5240	0.4903	0.4843
All: a0+a1+a2+a3	**0.6737**	**0.5690**	**0.5440**	**0.5073**	**0.5030**

Appendix IV: Results per category

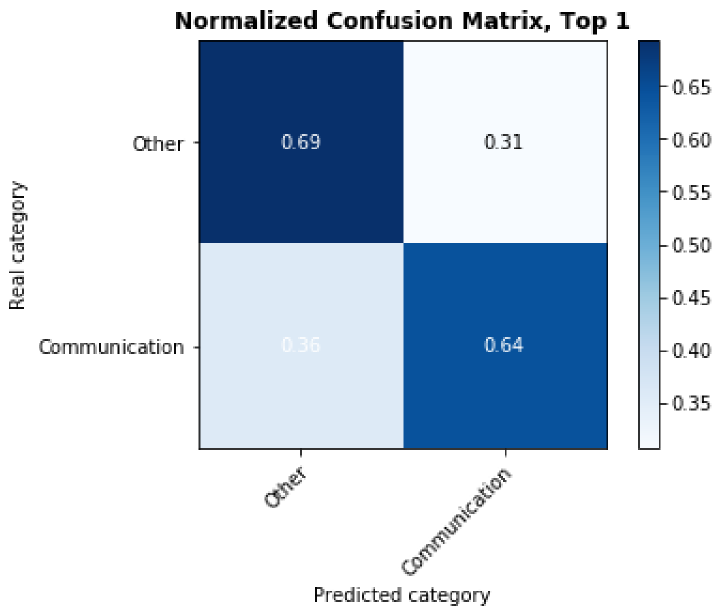
This appendix shows the classification reports and confusion matrices of every top *k* of categories. The color scale green-yellow-red is used in the classification reports, green meaning a high value and red a low one. The legend of the confusion matrix is besides the figure; a dark color means a large value and light a low one.

Top 1

Table 11. *Classification Report SVM Top 1*

	Precision	*Recall*	*Support*
Communication	0.59	0.64	1219
Other	0.74	0.69	1781

Figure 5. *Confusion Matrix SVM Top 1*

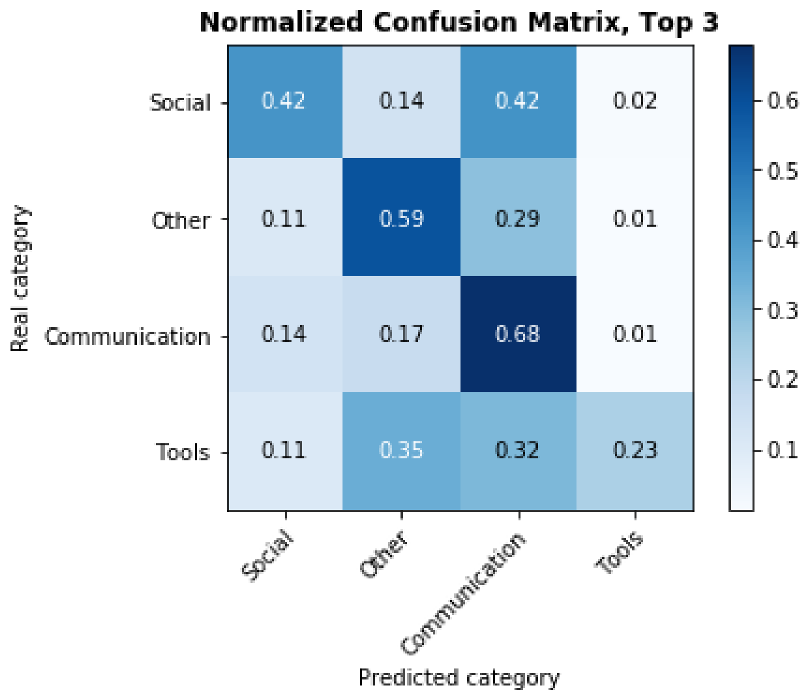


Top 3

Table 12. *Classification Report SVM Top 3*

	Precision	*Recall*	*Support*
Communication	0.58	0.68	1219
Other	0.62	0.59	997
Social	0.45	0.42	565
Tools	0.57	0.23	219

Figure 6. *Confusion Matrix SVM Top 3*

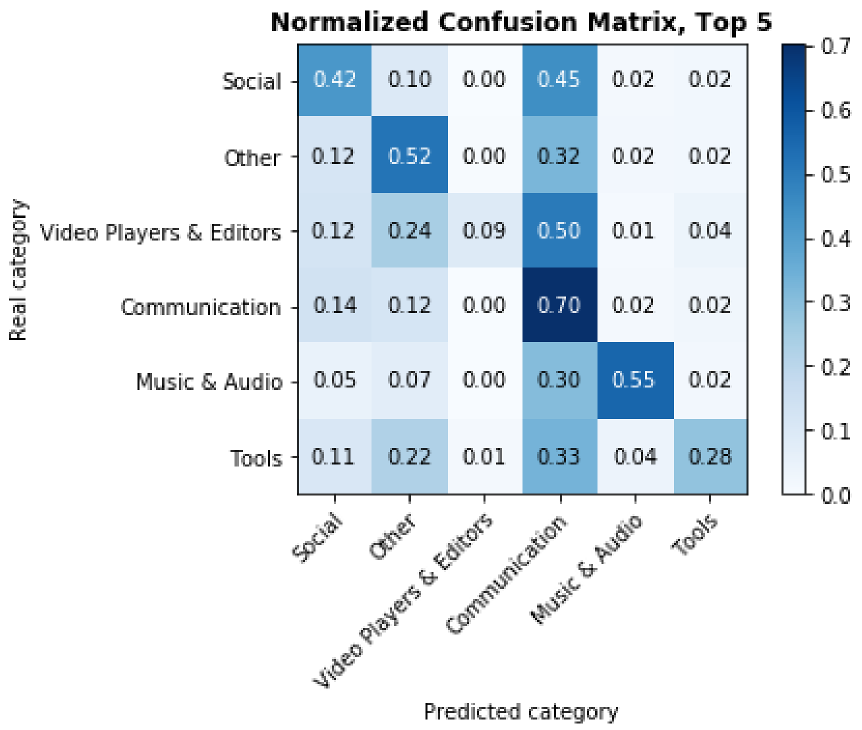


Top 5

Table 13. *Classification Report SVM Top 5*

	Precision	*Recall*	*Support*
Communication	0.56	0.7	1219
Music & Audio	0.64	0.55	164
Other	0.56	0.52	721
Social	0.45	0.42	565
Tools	0.51	0.28	219
Video Players & Editors	0.5	0.09	112

Figure 7. *Confusion Matrix SVM Top 5*



All categories

Table 14. *Classification Report SVM All categories*

	Precision	*Recall*	*Support*
Adventure	0	0	1
Background Process	0.17	0.03	74
Books & Reference	0	0	3
Business	1	0.12	8
Card	0	0	1
Casual	0	0	5
Communication	0.52	0.75	1219
Education	0	0	3
Entertainment	0.38	0.2	25
Finance	0	0	22
Health & Fitness	0	0	3
Lifestyle	0.36	0.28	36
Low Frequency App	0.55	0.37	191
Maps & Navigation	0.25	0.33	3
Music & Audio	0.64	0.55	164
News & Magazines	0.42	0.51	61
Personalization	0	0	8
Photography	0.52	0.26	100
Productivity	0.36	0.04	93
Puzzle	0	0	1
Racing	0	0	1
Shopping	0	0	10
Social	0.44	0.43	565
Sports	0.5	0.25	8
Tools	0.5	0.29	219
Travel & Local	0.53	0.44	54
Video Players & Editors	0.33	0.15	112
Weather	0	0	7
Word	0	0	3

Appendix V: Python code

All programming codes made in Python are visible in Jupyter Notebooks in the following link:

https://github.com/Jorina97/sharing-github.git