Data-Driven Marketing Analysis on Mobile Apps

BT3101 Business Analytics Capstone Project

NOVEMBER 7, 2016

Bao Jingxian	A0115420
Liu Yue	A0092394
Shen Lin	A0116453
Tang Jiahui	A0119415

Executive Summary

The recent rise in China’s smartphone adoption rate has resulted in a large amount of mobile data available, leading to the possibility of developing data-driven marketing techniques. This study aims to fill in the gap between state-of-the-art analytic frameworks and existing research for mobile application market in China. Specifically, we investigated the TalkingData data released in Kaggle Challenge and conducted user profiling using K-means algorithm. Based on the user profiles, an App Recommender System was constructed using the Apriori algorithm. Attribution analysis was conducted using Logistic regression. Finally, we made prediction on application installation rate for mobile device models.

Table of Contents

1. Introduction

China's smartphone adoption rate has reached 58% in Q1 2016. The annual growth rates were about 25% in past 2 years (PewResearchCenter, 2016). Besides the high and increasing penetration rate, mobile app market development has drove numerous mobile developers into the market. There are over 4 million mobile apps in China now and the number is still growing (Jourdan, 2016). As reported by eMarketer, a well-known marketing research company, an average Chinese smartphone user spends 24% of his time on his phone, but the media advertising expenditure on smartphone is only 8% in 2015 (SmartInsights, 2015). Hence, there are many unexplored opportunities in mobile marketing. In another review by Global Alliance of Data-Driven Marketing Associations, it was reported that 77% of marketing professionals believe in the prospects brought by data driven marketing (adWeek, 2015). Therefore, with in depth analysis in our project, we feel the urgency of highlighting the importance of mobile marketing and the vast opportunities existing in the field.

2. Data

2.1 Kaggle Challenge Data

The primary data source comes from Kaggle TalkingData Mobile User Demographics Case Competition (TalkingData, 2016).
Kaggle is a prevalent platform for case competitions related to statistical modelling and predictive analysis, where companies and researchers share their datasets globally for data scientists to contribute and discover the best model together.
The dataset of this project mainly comes from a challenge initiated by TalkingData. Being known as the largest big data service platform, TalkingData focuses on collecting user daily behaviour data on mobile devices. This dataset consists of anonymous users' demographics data, mobile devices properties, app usage and geolocation, which is collected from company's Software Development Kit (SDK) integrated with users' mobile applications. A more detailed overview of dataset could be view from the entity relationship diagram below. *gender_age* table contains user's demographic profile. *phone_brand_device_model* table is about the user's smartphone device information, which share the foreign device_id with the first table. *events* table is the events that collected from SDK in a time period of 7 days. So for each event of a device, it consists of a lot of *app_events*. With the last two tables, we know the category of each app.

2.2 External Dataset

2.2.1 App Category Data

We also used Beautifulsoup package in python to extracted external dataset in iTunes Store (2016), which includes top 100 free apps and top 100 paid apps downloaded by global users. After the name of the app, the website also gives the category of that app.
The dataset could be imported into RStudio to view a summary of category. It shows the distribution of app categories, which could be used for future references or further dimensionality reduction.

2.2.2 Zhongguancun Online

Data for features of device models is obtained from Zhongguancun Online (Zhongguancun, 2016). It is a e-commence website and a online portal for all kinds of hardwares. For mobile phones, it provide comprehensive information on devide level with each device having its own webpage.

3. User Profiling

Profiling for smartphone users is to group those with similar demographic characteristics and app usage behaviours together and identify profile for each group. This facilitates marketers in developing a deeper understanding on their users.

3.1 Literature Review

Although user profiling has gained its attention long ago, its application in the mobile context is still quite new. In this section, 2 papers done by previous researchers will be discussed to help us develop our own understanding and approach towards the problem.

The paper "Smartphone's Customer Segmentation and Targeting" (Hamka, 2012) studies the market segmentation on the perspective of application provider based on mobile app usage. In the area of market segmentation, there are 4 commonly used dimensions suggested by Kotlet (2003), which are demographic, psychographic, geographic and behavioral segmentation. With reference to these 4, Hamka included demographic, psychographic and behavioral dimensions into his model.

He used Latent Class Analysis (LCA) as the method, which is to find the probability of an observed data belonging to a latent class. Due to the fundamental assumption – local independence of LCA, Hamka conducted correlation analysis among observed variables to eliminate highly correlated variables. Hence, he had 3 observed variables which are average app run per day, average app installed per day and average url browsed per day, and 2 covariate variables which are demographic and psychographic attributes for the model. The observed variables are used to run the Latent Class Cluster Model while covariate variables are used to support resulted clusters. Last, he got 6 clusters from 3 observed variables.

In the paper “Intelligent Mobile User Profile Classification for Content Personalization” (Paireekreng & Wong, 2009), instead of LCA, Cluster analysis was preferred. As compared to LCA, Cluster analysis has a much smaller complexity, fewer assumptions to fulfil and more widely used in profiling research area. Though Hamka provides us lots of insights into variables selection, Cluster analysis suggested by Paireekreng and Wong is adopted in this problem.

3.2 Problem Statement

The motivation of doing user profiling is to allow marketers to do targeted advertising and to communicate with app users in a more personalized way. So knowing the importance of predicting user profiles to marketers, we propose our first question: Can we identify user profiles from their daily app usage and basic demographic information?

3.3 Hypothesis

H1$_0$: Sample users can be divided in subgroups in their demographic and usage behaviors of downloaded apps.

H2$_0$: There are statistically significant differences between the groups in the structure of their active rate on business related apps.

H3$_0$: There are statistically significant differences between the groups in the structure of their active rate on baby care related apps.

H4$_0$: There are statistically significant differences between the groups in the structure of their active rate on game related apps.

If the above 4 hypotheses are not rejected, then we will give tags to these clusters as “business professionals”, “baby care population” and “active gamers” accordingly.
4 variables which are age, gender, phone price and active rate of each app category are included into the model. Phone price is an indirect indicator of a user’s purchasing power or income. As suggested by Hamka (2012), age, gender and income are common demographic attributes used for market segmentation. Active rate of each app category basically measures the number of times a user is actively using apps of a particular category. In Hamka’s paper, he used average number of app run to indicate a person’s degree of mobile app usage. With reference to this idea, we include average active rate of each app category to indicate the user’s degree of app usage of a particular category.

3.4 Method

3.4.1 ETL Overview

Age and gender can be obtained directly from the table *gender_age* of Kaggle dataset. Phone price is scraped from ZOL website. Active rate of each app category can be obtained through several ETL processes of Kaggle dataset.

By joining *events* and *app_events* tables on foreign key *event_id,* we are able to obtain all app events of a *device_id.* Then by joining *app_events* with *app_labels* and *label_categories*, we will get a giant table with each row representing an *app_event* of the device, knowing its app category and active state. Last, to calculate a device's active rate of each app category, we just need to sum up *is_active* attribute of every *app_events* of the device under a particular category.

3.4.2 Algorithm

To decide on the clustering algorithm will be used, 3 commonly used clustering models, which are hierarchical clustering, k-means clustering and DBSCAN clustering are analyzed. Comparison is made based on objective functions, complexity, parameter and suitability to small and large dataset as shown below (Abbas, 2008 & Vijayalaksm, 2012).

Hierarchical	**K-means**	**DBSCAN**
metric and linkage criteria	minimize sum of squares	size of reachable points
high complexity	smaller complexity	small complexity
no k required	set k in advance	no k required
small dataset	large dataset	small dataset

There is no objectively "correct" clustering algorithm, the validity of a clustering is in the eye of the beholder. Also though there are some limitation in traditional algorithm, modified method allows more flexibility. For example, standard k-means algorithm is not directly applicable to categorical data, but its variation allows clustering with a mix of categorical and numeric data. So by only considering the dataset size and complexity, K-means is adopted for this problem. If K-means does not give desirable results, other clustering algorithms would also be tried.

3.4.3 ETL

To obtain user's' demographic information and device app category, external datasets are introduced. Phone price of each device can be obtained from 2 csv files crawled from ZOL and the category of each app can be obtained from our app clustering result. Hence, in total 7 csv files are processed to get the final dataset for clustering.

The entire ETL process can be summarized into 3 steps. The first is data cleaning. It was done by dropping duplicates and null, reading selected variables, renaming variables and aggregating.

The next step is joining csv files. I used colours to indicate keys shared among different csv files. By identifying keys, files were merged together. Hence after all merging processes, two merged data frame are obtained. One contains device_id and the user's demographic information, which are price, gender and age. The other one contains device_id, the category of an app on the device at specific time as well as the active status of app.

csv files	Variables
events.csv	event_id , device_id
app_events.csv	event_id , app_id , is_active
relabel_apps.csv	app_id , cat
gender_age_train.csv	device_id , gender , age
device_model_count.csv	index , brand , model
specification.csv	index , price
phone_brand_device_model.csv	device_id , brand , model

In the last stage, the focus is the aggregation as well as creating pivot table. I aggregated on device_id and category to calculate out the active rate of the device on each category. Then by creating a pivot table on app category, all app category names became new column names. The resulted data frame was merged with the data frame containing demographic information to obtain the final dataset.
So, in total, the final dataset has 22,686 records and contains 17 variables. Except the device_id, 16 variables are used in the clustering model. Among these 16 variables, 15 are numeric and 1 is categorical, which is gender.

3.5 Results

3.5.1 Model Selection

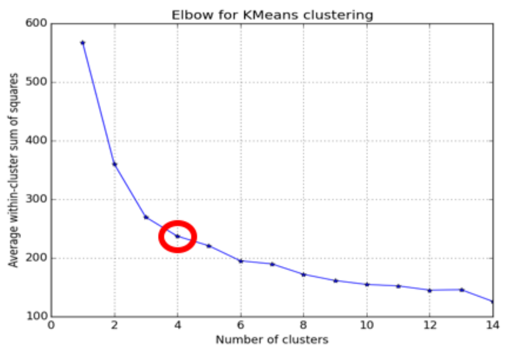
As mentioned above, k-means as well as its variation is the clustering algorithm. Optimal k could be found using elbow method. Decision tree is also used to assist the interpretation of formed k clusters.

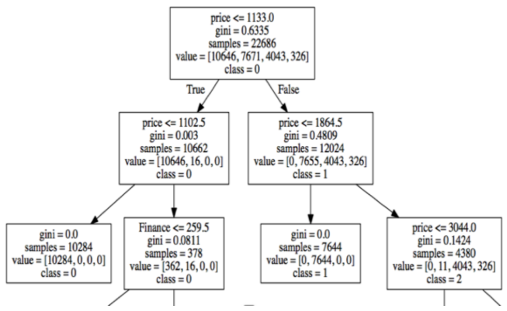
k-prototypes

k-prototypes is a variation of k-means, published by Huang in 1997. It is also a function in Python k-modes package. It removes the limitation on numeric data. So it is able to cluster mixed numeric and categorical data. Though with this advantage, it is still not the ideal model for our case. It has low efficiency, also there is limited literature reviews to refer. Most importantly, there is no clear elaboration on the value of one parameter, hence it is difficult to justify our choice and make sound of the analysis.

KMeans

KMeans is a clustering function provided in Python scikit-learn package. To solve limitation on numeric data, the categorical attribute gender is transformed into a binary data, where 1 represents female and 0 represents male. By calculating the average within-cluster sum of squares and plotting it against number of clusters, I identify 4 as the optimal k.

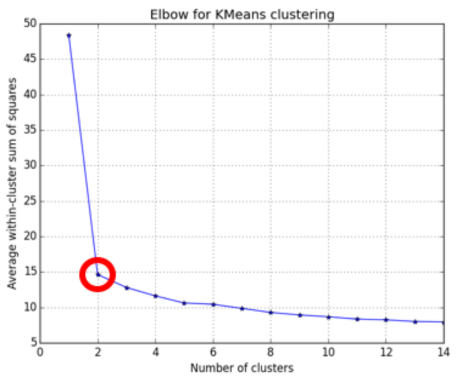


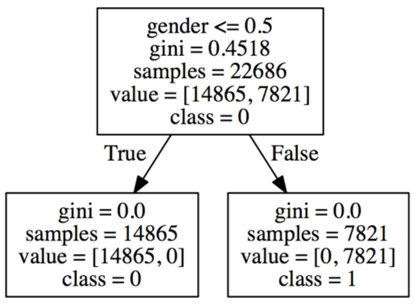


However, in the decision tree plotted, price is recognized as the dominant attribute. This is mainly because price has large magnitude compared with other attributes, so it dominates in the cost function (Euclidean distance) and makes result biased.

KMeans – scaled dataset

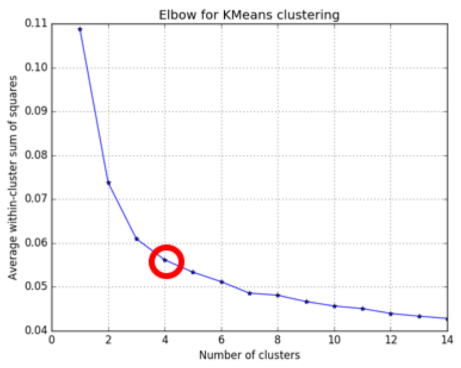
Knowing the limitation on the original dataset, each variable is scaled accordingly by calculating the ratio of difference with the minimum value to the largest possible difference in the column. The elbow plot suggests 2 clusters for this case. However, 2 is too few for the dataset with 16 variables and 22686 records. The decision tree supports our view as the clustering is merely based on one variable, gender.





KMeans – normalized dataset

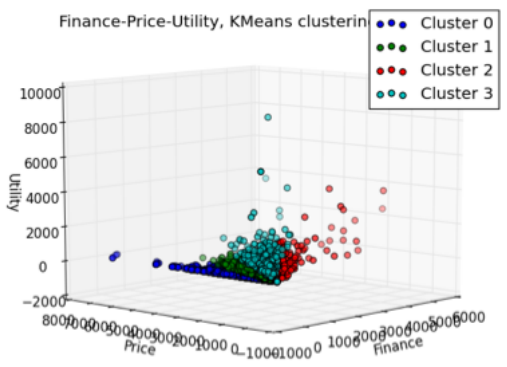
As normalization is also one recommended method for unscaled dataset, Python numpy package is used to normalize the original dataset. The elbow method suggests optimal k as 4. Also the top attributes in decision tree are utility, price and finance.

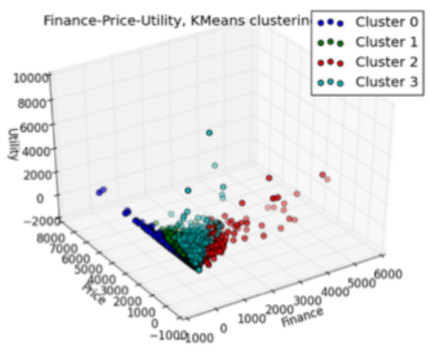


Hence, KMeans on normalized dataset is the final model for our user profiling.

3.5.2 Findings

By using Python matplotlib package, two 3-D graphs are plotted to visualize decision tree result.





It is clearly shown that when plotting based on finance, price and utility, elements in each cluster are closely packed together. Also characteristic of each cluster in terms of finance app usage, utility app usage and device price are identified as below.

- Cluster 0: wide range phone price, low finance activity, low utility activity

- Cluster 1: wide range phone price, medium finance activity, low utility activity
- Cluster 2: medium range phone price, wide range finance activity, large range utility activity
- Cluster 3: medium range phone price, medium finance activity, large range utility activity

Interpretation of clustering result is subjective as it changes when the focal variables vary. In this case, we only examine the characteristics of each cluster when focal variables are phone price, finance app and utility app usage. It is just one aspect of the entire comprehensive picture of the clusters. These three are chosen because the decision tree suggests them as the top classification attributes.

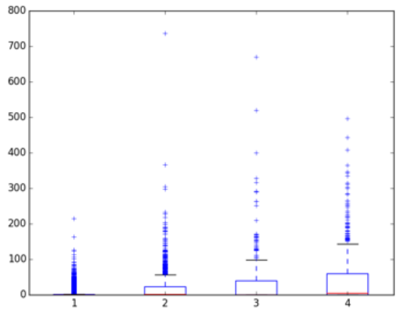
3.5.3 Hypotheses Examination

With the understanding on the final model as well as the result, hypotheses proposed could be examined.
$H1_0$: *Sample users can be divided in subgroups in their demographic and usage behaviours of downloaded apps.*

True. It is supported by the above 3-D plot. 4 clusters are formed and explained well using phone price, finance app usage and utility app usage.

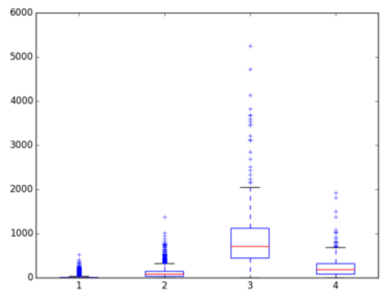
$H2_0$: *There are statistically significant differences between the groups in the structure of their active rate on business related apps.*

True. It is supported by the boxplot of finance app active rate of each cluster. It shows that in general cluster 0 has relatively lower active rate and cluster 3 has relatively higher active rate.



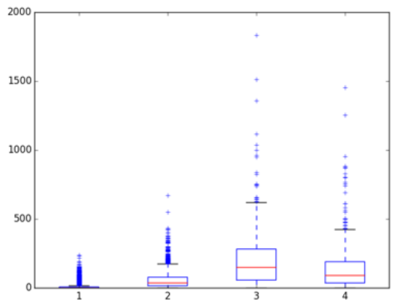
$H3_0$: *There are statistically significant differences between the groups in the structure of their active rate on baby care related apps.*

True. It is supported by the boxplot of finance app active rate of each cluster. It shows that there is significant difference among the 4 distributions. In general, cluster 0 has relatively lower active rate and cluster 2 has relatively higher active rate.



***H4$_0$**: There are statistically significant differences between the groups in the structure of their active rate on game related apps.*

True. It is supported by the boxplot of finance app active rate of each cluster. It shows that there is significant difference among the 4 distributions. In general, cluster 0 has relatively lower active rate and cluster 2 has relatively higher active rate.



3.6 Conclusion

In conclusion, the user profiling study of this project serves as the further study on *Smartphone's Customer Segmentation and Targeting (Hamka, 2012)* which only focusing the user profiling based on general mobile app usage. In this paper, detailed app category and respective active rate are included to perfect the research area.

So after applying Python KMeans function on normalized dataset, 4 clusters are observed, result supports hypotheses and proves the significant statistical difference in the structure of active rate in finance, baby care as well as game app among clusters.

However, in this study, the cluster of a user could only be identified using several variables, such as phone price, finance app usage and utility app usage. We recognize that it is hard to identify a cluster merely based on one single app category activity because distributions of single app category activity among clusters are not explicitly exclusive. Hence, to facilitate further study based on the user profiles, users in the 80 percentile of a particular app category are identified as active user in that category.

One possible limitation of this study is short duration of entire dataset. The Kaggle competition only provides one week data, which may put limitation on our model and analysis. User's behaviour on mobile app may not be correctly reflected by the one week data provided.

To conclude the discoveries and findings in user profiling part, users' demographic information as well as mobile app usage indeed are able to segment smartphone users into clusters. In the future, this study could be further enhanced by using data with longer duration.

4. App Recommender System

After customer profiling, there's also potential for exploring correlations between app categories. The relationship could allow advertisers to put targeted advertisements between smartphone apps, and further improve advertiser's market campaign performance. It is similar as the idea of a recommender system.

4.1 Literature Review

Recommender system for mobile apps is a platform to provide users with potential app suggestion based on individual preferences. It aims to solve the increasing challenge for users to target apps of their interest and also provides a solution for data-driven marketing such as advertisement campaign between apps. To get more understanding of this area, several research papers in recent five years have been reviewed as references.
Ricci, Rokach & Shapira (2011) have pointed out in their book *introduction to recommendation system* that the approaches to build a recommender system mainly consist four sections, which are collaborative filtering, content-based filtering, hybrid recommender systems and personality-based recommender systems respectively. A more detailed summary of literature review regarding this four approach is listed in table below.

Approaches	**Dependent Variable**	**Pros & Cons**
Collaborative Filtering	past behavior similar decisions by other user	Requires large number of information on users' behavior, activities or preferences. Issue of cold start, scalability, and sparsity. (Lee, S., Yang, J. & Park, S., 2007.)
Content-based Filtering	Similar properties between items Characteristics of items	Needs smaller set of user info to get started but the diversity is also less. It is limited in scope as it can only make recommendations that is similar to original seeds
Hybrid System	Combine two approaches above	Combine two approaches, more effective and higher accuracy
Personality-based Approach	User's personality	A new approach requires social media data to predict user personality (Ricardo, B., 2016)

In another research paper published by Xia, Wang & Zhou in 2011, it indicated for recommender systems between mobile apps, Google plays utilizes collaborative filtering approach and they further evolved it to a more diversity measurement-based framework. Linking back to our project's dataset, we have more user specified data and less application level data except for categories as independent variable. Similar as Google Play, a collaborative filtering approach would thus be more suitable for us to achieve diversified recommendation.

Besides, a research that relates back to advertisements among smartphone apps adopted an unsupervised learning approach and utilised association rule (Reps J., Aickelin U. & Garibaldi J., 2015). K-means clustering algorithm is utilised at first stage to segment user groups, which is similar as customer profiling in our project. After that, Apriori algorithm of association rule mining is conducted on a dataset collected in 2014 consisting 60969 log files over 2 million users globally. More than 100 rules of finance, lifestyle and entertainment field are discovered. It is believed that these frequent trends could be utilised to improve future performance of marketing campaign in advertisement.

4.2 Problem Statement

As described in previous sections, exploring correlations between app categories is the fundamental for building this recommender system. The motivation for doing analysis on this topic is to provide data driven recommendations for advertisement campaign. And thus the target audience under this section would be advertisers or relevant advertising companies that put inter-app advertisements.
The problem statement thus would be whether there is any specific correlation between app categories in different user groups.

4.3 Hypotheses

In order to provide recommendations for app advertisement, we would like to find some general trends between app categories for different user groups. The corresponding hypotheses for examining whether there's specific correlation between app categories would include the following:

- **$H1_0$**: for professional's user group, social and finance categories of apps might be highly correlated. For the people who uses social apps, he is more likely to use finance related apps as well.
- **$H2_0$**: for student's user group, there is no significant relationship between Game and Educational apps, meaning we cannot find {"game" => "education"} as a top rule in student group.
- **$H3_0$**: for all user groups, there is a positive relationship between travel and weather categories of app. People who use travel related apps frequently might also be interested in apps under weather category.

The primary variables involved in this section is *device_id*, which represents users. After clustering result is generated from customer profiling, a user group label will be assigned to each device. And thus *user group* becomes another important variable for us to partition matrix. And under each user record, the variable is *app category*. However, as the analytical model used in this section is unsupervised learning approach, there will be no independent or dependent variable involved.

4.4 Method

4.4.1 Algorithms

As mentioned in literature review, Collaborative Filtering (CF) has been proven to be a more feasible methodology for our dataset. It has two primary forms, which are user-based CF and item-based CF respectively. Overview related to these two methodologies are summarized in the table below (Herlock, Konstan, Terveen & Riedl, 2004).

Methodologies in CF	**Main Idea**	**Popular Algorithms**
User-based collaborative filtering	Look for users share same rating pattern Use the past ratings to predict future action of active user	User based nearest neighbor algorithm k-nearest neighbor (k-NN) method
Item-based collaborative filtering (users who bought x also bought y)	Build on item-item matrix between pairs of items Infer the taste of current users by examining matrix and matching user's data	Slope-one algorithm Pearson product-moment Correlation coefficient Association rule Apriori Algorithm

Based on the data we have, which are mobile phone apps and its categories, an algorithm to the per user unit case is infeasible, as our dataset lacks user level rating regarding each application. What we could generate through our dataset are different matrices containing individual user's installed app categories, with each line indicating one record.
Referring to previous work related about advertisements between smartphone apps, the unsupervised association rule under item based collaborative filtering algorithm is more suitable for building app category matrix for each user group (Reps, Aickelin & Garibaldi, 2015). Apriori rule is also the most commonly used analytical model for discovering "*what category goes with what*". Therefore, Apriori algorithm for association rule will be the potential analytical model for this project to explore for recommendations and correlations between app categories.

4.4.2 Data & ETL Process Overview

In order to apply association rule algorithm, we need transform our data about user and the categories of apps they used, in order to build binary matrices for different user group.

Each row in this binary matrix represents a user record under this user group; each column represents an app category. A full list of categories will be listed as different columns. If the user installed any app under this category, a 1 will be put in the corresponding place in matrix, otherwise it will be 0.
And three measurements which are support, confidence and lift will then be calculated for different sets of app categories and rank them accordingly to generate top rules.
For getting this binary matrix, a ETL process containing 5 steps should be conducted.

- Step 1: Extract

Except for Kaggle dataset, we also scrapped external data source from apple store about its app category data, which could be used for further references or future dimensionality reduction purpose.

- Step 2: Filtering

We only need to get one set of installed application data for each *device_id*. However, events dataset contains lots of apps data for same device at different timestamps.
We need to extract the last occurrence of each device in event dataset to get a full list of apps installed by one user. Thus, we reversed the data frame, and remove duplicates in R to get a reduced dataset.

- Step 3: Join table

App Events dataset was then joined together with events dataset using *event_id* as a primary key. The labels are joined together to each app using *app_id* as primary key.

- Step 4: Aggregation

A large dataset is generated through joining tables, where one device has several apps installed and its corresponding *label_id*. In the next step, we are going to aggregate device and its apps to be a binary matrix described before.

- Step 5: Labeling

At last, for interpretation purpose, we will relabel for app categories and each app, and also attach user group result given by clustering analysis to each device. This process makes the whole dataset meaningful and easier for interpretation.

4.4.3 Expected Findings

In different user group, the expected finding would be some top trends ranked by three measurements. For example, if ranking by confidence, there will be some top rules between app categories in particular user group. (e.g.: Professionals: “Social” => “Finance”, with highest confidence 0.97)

4.5 Relabelling Categories

In this section, categories relabeling process will be discussed in detail, as it is a step in ETL that is particularly critical. An accurate relabeling process serves as the basis of all parts of analysis.

In the dataset, there’s a massive category dataset including more than 900 categories, which are unstructured text data. It’s impossible to analyze directly based on categories from this large set of categories. Some possible shortcomings include the inference appears too trivial, or the matrix we build for analysis can be too sparse. So we need to relabel and classify all 900 +, and also give all the apps a new label. This could not only provide a basis for my analysis but are also the fundamental of other part of analysis.

4.5.1 Approach

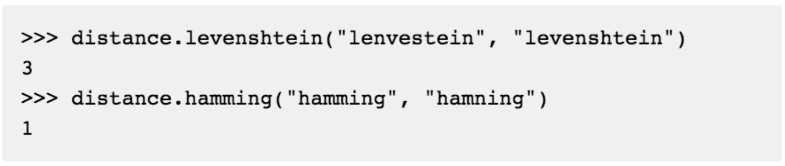
There are three primary proposals of relabeling approach, each of them has its own pros and cons.

- *Approach 1:* Directly manual relabel all categories. However, it might be too objective. It involves a large amount of workload. Besides, at least three people are required to work independently to make judgements.
- *Approach 2*: Semi manual approach by replacing substring. For instance, we could replace all categories containing ‘*game’* substring to be “Game” category. It relatively reduces workload. However, lots of categories don’t have an obvious substring, which leads back to approach 1 of manual relabeling.
- *Approach 3*: Clustering method. This method clusters original categories into sub cluster groups, and aims to relabel all categories in a same cluster by manually relabeling. It is less subjective, and more accurate. It involves less manual work as we only need to relabel N clusters result. However, there are some more complicated technical considerations, which will be covered in later part.

After comparing I decide to proceed with approach 3 as our major approach for proceeding to relabeling process.

4.5.2 Defining Similarity Matrix

In order to put the original categories data into clustering algorithm, a similarity matrix needs to be constructed for pairwise categories data.
For defining similarity matrix, I first explore online packages for constructing similarity matrix for text data.



```
>>> distance.levenshtein("lenvestein", "levenshtein")
3
>>> distance.hamming("hamming", "hamning")
1
```

An example of online package for calculating distance

It offers hamming, levenshtein distance algorithm. However, it calculates the similarity between two categories by only comparing differences between strings. And it gives no consideration about semantics of categories or the meaning of the word.

Thus, an alternative self-defined matrix would be better for evaluating distance and similarity between categories.
Because an app can fell into many categories, a matrix of category-category similarity can be built by calculating function f, where

$$f(a,b) = \frac{\{intersection\ of\ apps\ fall\ in\ two\ categories\}}{\{union\ of\ apps\ fall\ in\ two\ categories\}}$$

It gives a matrix with each item range within [0,1].

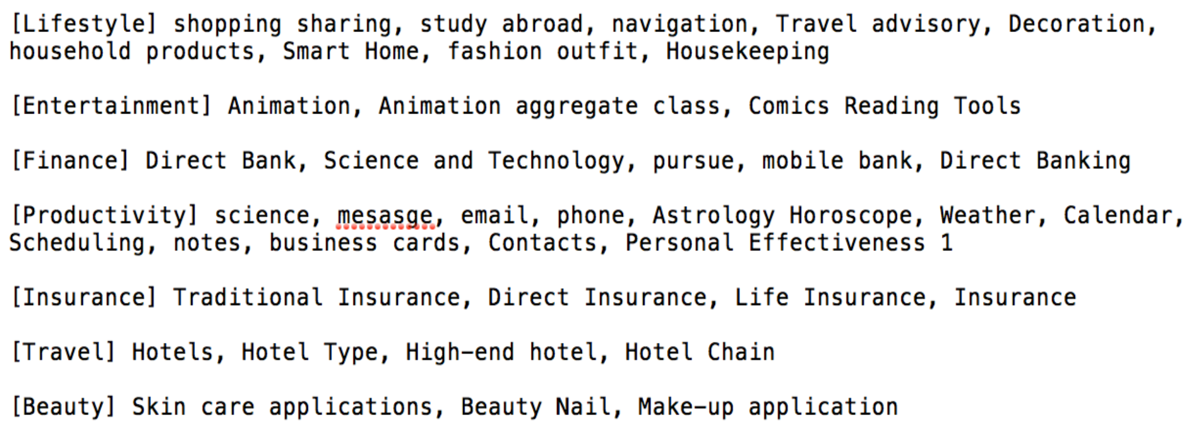
4.5.3 K-means Clustering Group parameter selection

After a matrix is obtained, the next step is to select a suitable k-mean parameter for determining the number of clustering group.
In the first trial, the parameter is set to be 20. However, the result shows several big clusters with a large chunk of categories. And some clusters can be a mixture of various categories and are hard to summarize. Besides, there are not so many variants of relabeled categories, which are only 7 categories.

Thus, we move on to change the parameter of group to 40 groups. This time it performs much better than the previous model. It offers several smaller clusters that has obvious common label appeared. And it is easier to summarize and manually label all clusters by three independent judgers. Furthermore, more variants of relabeled categories are obtained, which are 14 groups. Therefore, we decided to utilize this version of relabeling as our ETL result.

4.5.4 Relabelling Accuracy Evaluation

The clustering result is hard to evaluate quantitatively, but can only reviewed in a qualitatively way. By looking into a snippet of result, it can be shown that the relabeling is quite intuitive and accuracy is acceptable.



```
[Lifestyle] shopping sharing, study abroad, navigation, Travel advisory, Decoration,
household products, Smart Home, fashion outfit, Housekeeping

[Entertainment] Animation, Animation aggregate class, Comics Reading Tools

[Finance] Direct Bank, Science and Technology, pursue, mobile bank, Direct Banking

[Productivity] science, mesasge, email, phone, Astrology Horoscope, Weather, Calendar,
Scheduling, notes, business cards, Contacts, Personal Effectiveness 1

[Insurance] Traditional Insurance, Direct Insurance, Life Insurance, Insurance

[Travel] Hotels, Hotel Type, High-end hotel, Hotel Chain

[Beauty] Skin care applications, Beauty Nail, Make-up application
```

A snippet of relabeling result

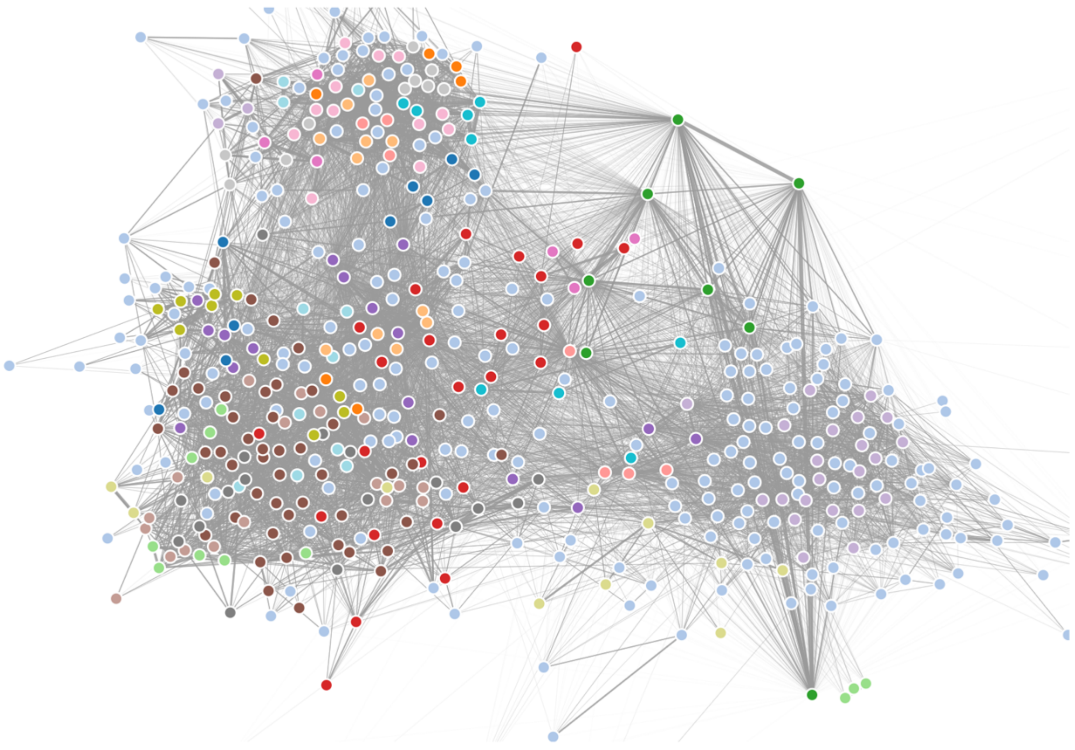
For example, all hotels are clustered together and mapped to *travel* category. And skincare, beauties are clustered in one group for us to manually relabeled them to *beauty* category. All the clustering result makes sense and is of acceptable accuracy in this way.

4.5.5 Visualisation of Clustering Result

Besides, a d3 visualization of clustering result is also built to visualize the clustering result of 40 groups, in order for us to have a better understanding of the result geometrically.

However, we only know pairwise similarity between categories as defined before, and all categories are text data. There is a problem that the categories have no geometric definition for us to map them into 2D plane as they are just high dimension text data.

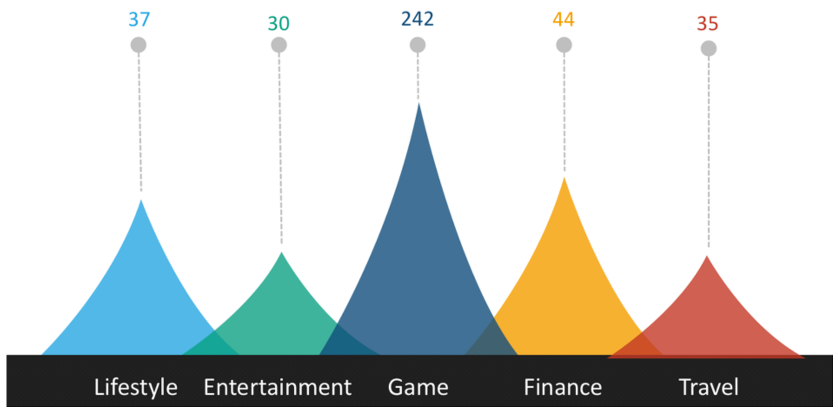
Force directed graph algorithm are selected to visualize the clustering result in this way. It draws link using forces between two nodes. The similarity could represent forces between categories, with a higher similarity score indicating a stronger force that pulls two categories' node together (Fruchterman & Reingold, 1991)



d3 visualization of clustering result

The graph above shows the clustering result visualization. It has 40 colour indicating 40 clustering groups, showing some patterns of gathering of same colour in 2D space together.

Thus, linking back with the detailed text evaluation, we could conclude that the relabelling clustering result is of acceptable accuracy. And we could proceed with this new dataset with relatively high confidence.



Top five categories relabeled

The figure above shows a descriptive analysis of relabeled categories. Top five categories relabeled are lifestyle, entertainment, game, finance and travel.

4.5.6 Relabelling category for each app & Build binary matrix

Next steps in ETL is to add app label to each app. As one app can fall in various categories before processing, I tag each app by taking the most frequent and common occurred categories after relabeling result. After that, the tagged app data is added back to joint dataset. Aggregation and filtering that are similarly to ETL process in Section 3 are utilized.

The user group tag given by Section 3's part of analysis is added to overall dataset. Categories appeared in one device are combined in one dataset for each user group, so that we can build the final binary matrix to do association analysis.

$$\begin{bmatrix} & app\_category\_1 & app\_category\_2 & \dots & app\_category\_n \\ device\_id\ 1 & 1 & 1 & 0 & 1 \\ device\_id\ 2 & 0 & 1 & 1 & 0 \\ \dots & .. & \dots & \dots & \dots \\ device\_id\ n & 0 & 0 & 0 & 1 \end{bmatrix}$$

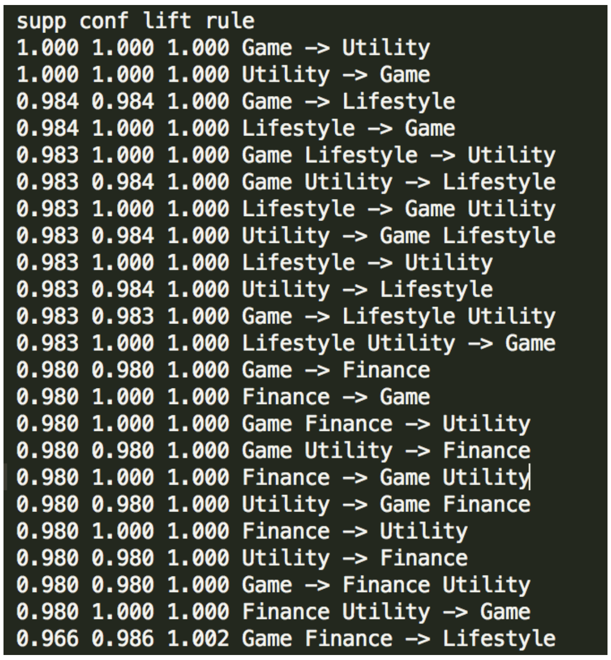
Each row in this binary matrix represents a user record under this user group; each column represents an app category. A full list of categories will be listed as different columns. If the user installed any app under this category, a 1 will be put in the corresponding place in matrix, otherwise it will be 0.

4.6 Analytics Result

After the binary matrix for association rule is built, we put the matrix into Apriori algorithm to do the association analysis. And top rules are ranked by support, life and confidence three measures. Besides, only the most significant top 1% rules are extracted for insights.

4.6.1 Result Overview

Several trends can be discovered by ranking the association results by support, confidence and lift for each user group. An example of 'game lover' user group's result can be shown in the file below, ranking by support measurement.

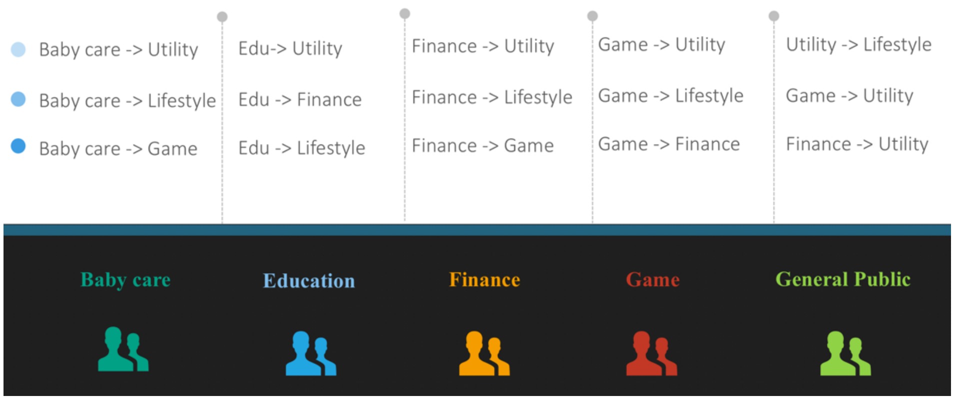


```
supp conf lift rule
1.000 1.000 1.000 Game -> Utility
1.000 1.000 1.000 Utility -> Game
0.984 0.984 1.000 Game -> Lifestyle
0.984 1.000 1.000 Lifestyle -> Game
0.983 1.000 1.000 Game Lifestyle -> Utility
0.983 0.984 1.000 Game Utility -> Lifestyle
0.983 1.000 1.000 Lifestyle -> Game Utility
0.983 0.984 1.000 Utility -> Game Lifestyle
0.983 1.000 1.000 Lifestyle -> Utility
0.983 0.984 1.000 Utility -> Lifestyle
0.983 0.983 1.000 Game -> Lifestyle Utility
0.983 1.000 1.000 Lifestyle Utility -> Game
0.980 0.980 1.000 Game -> Finance
0.980 1.000 1.000 Finance -> Game
0.980 1.000 1.000 Game Finance -> Utility
0.980 0.980 1.000 Game Utility -> Finance
0.980 1.000 1.000 Finance -> Game Utility
0.980 0.980 1.000 Utility -> Game Finance
0.980 1.000 1.000 Finance -> Utility
0.980 0.980 1.000 Utility -> Finance
0.980 0.980 1.000 Game -> Finance Utility
0.980 1.000 1.000 Finance Utility -> Game
0.966 0.986 1.002 Game Finance -> Lifestyle
```

Support shows how frequent these two categories appears together in the database. For instance, the top rule in this file indicates that for all data in game user's device record, 100% of the records show that the user uses *game* and *utility* category of apps together.
Confidence is an indication of how often the rule has been found to be accurate. The top rule again indicates the proportion of the users that uses *Game* also uses *Utility* is 100%.
Lift of a rule is defined as $Lift\ (x \to y) = \frac{\text{support}\{\text{x,y}\}}{\text{support}\{\text{x}\}*support\{y\}}$, which shows the ratio of observed support to that expected if X and Y are independent.

All of them can be a good measure of how significant the rule is. For convenience, support is used as ranking criterion in the following analysis, as it gives a nicer data formatting.

For each user group, top three rules ranked by support are summarized below.



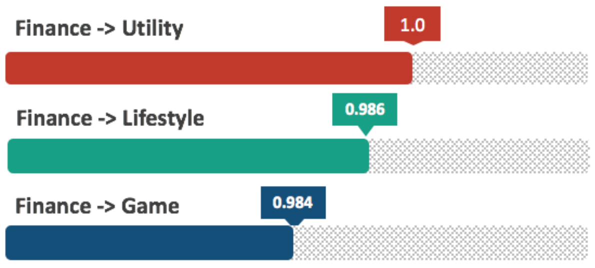
The top 3 rules for five user groups.

The result indicates that for each user group, the top three rules most likely to link with utility and the categories of that group. For instance, finance professional user groups tends to use finance apps with utility, lifestyle and game apps together.

4.6.2 Hypothesis Examination

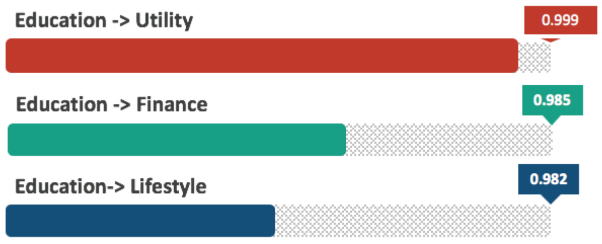
The examination of previous mentioned three hypotheses for this section are listed below.

● **H1$_0$**: for professional's user group, social and finance categories of apps might be highly correlated. For the people who uses social apps, he is more likely to use finance related apps as well.



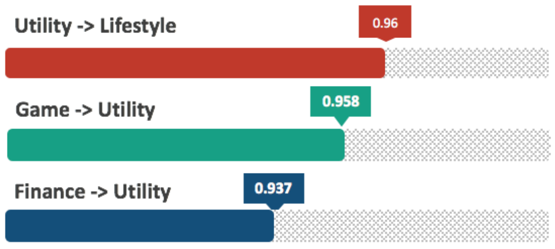
The result given by our analysis shows that this hypothesis is **False.** There's ***no obvious relationship*** between social and finance. The top correlated categories for this user group are finance to utility, lifestyle and game.

● **H2$_0$**: for student's user group, there is no significant relationship between Game and Educational apps, meaning we cannot find {"game" => "education"} as a top rule in student group.



True. There's ***no correlation*** between game and education. The top correlated categories for this user group are education to utility, finance and lifestyle

- **H3$_0$**: for all user groups, there is a positive relationship between travel and weather categories of app. People who use travel related apps frequently might also be interested in apps under weather category.



This hypothesis ***cannot be tested*** due to lack of weather label in ETL relabeling process. Weather is actually classified as a subcategory under Travel clustering in our relabeling process, which in some way indicates that these two categories are highly correlated. However, this hypothesis still cannot be tested using our association rule result.
The top correlated categories for general public are utility to lifestyle, game to utility and finance to utility.

4.7 Conclusion

Overall, in this section, we found that for different types of user group, there are several top correlated rules between mobile categories. We also discussed several interesting ETL approaches in pre-processing the data and relabel unstructured text data of categories.

Some counter-intuitive findings are discovered in our result as well. There are some categories of apps that we think are correlated together by common sense, which may not hold to be true.

Besides, in our analysis result, it can be noticed that utility shows a pattern of high importance. One conclusion can be draw from this phenomenon is that putting ads to utility apps might be effective for all kinds of apps. However, this could also reflect a constraint of our section. Utility apps could be a confounding factor, because they are pre-installed by default in their mobile devices and users are unable to delete them. To further investigate the effect of utility category, a more detailed data is needed to show whether the apps are self-installed or forced by device settings.

One possible limitation of this analysis is that the result of our model cannot be quantitatively examined. The top rules mined by association rules are some hidden correlations between categories, and three measures can be used to rank the importance of all the rules. Yet the effectiveness and accuracy of the rules cannot be tested as there's no way to verify it quantitatively.

To conclude the discoveries and findings in app recommendation part, these rules show some certain pattern of correlations between different app categories. Thus, the business application for this section could have a great impact on marketing strategies. For example, advertisers could utilize the findings in this section to develop their advertisement campaign or show inter-app ads for optimizing their publicity effect. Besides, affinity marketing can be conducted by displaying promotions and references codes in corresponding categories of apps to attract more population.

5. Attrition Analysis

Attrition or churn refers to the loss of customers. It helps understand the behavior of users who are most likely to discontinue the use of the app and identifies them before they quit in order to plan customer retention strategies better.

5.1 Literature Review

The concept of attrition analysis is only recently adopted in mobile application contexts, but has been studied in a variety of disciplines for decades, such as retail banking, insurance, and telecommunication. Hung, Yen, and Wang (2006) applied various data mining techniques to predict attrition for each subscriber of a telecom operator. In the paper, customer demographics, billing information, contract/service status, call detail records, and service change log are used as features; decision tree and neural network techniques have been utilized to predict models.

In another work, Coussement and Van (2009) improved the customer attrition prediction by integrating emotions from client/company interaction emails using several machine learning methods. In the study, logistic regression, support vector machines and random forests classification methods are used to define churners and non-churners classes.

In the field of mobile game analytics, Hadiji, Sifa, and Drachen (2014) present the first cross-game study of churn prediction in Free-to-Play games from Europe. Four different classification algorithms, namely, Logistic regression, Naive Bayes, Decision Trees, and

Neural Networks are utilized to predict churners by using number of sessions, number of days, current absence time, average playtime per session, average time between session, premium user flag, predefined spending category, number of purchases and average spending per session. In the study, the decision tree algorithm has found to reach the highest F1 score of 94.5.

5.2 Problem Statement

Can characteristics outside app predict churners for a certain type of apps?
App marketers focused on attracting as many new users to the app as possible. However, if the installed user chooses to abandon the app, all the money, manpower, and time for marketing will be wasted. In fact, 25% of the users will only use an app once. For mobile games especially, only 10% of the users will stay after three months on average (Localytics, n.d.).

On the other hand, the previous paper on mobile app attrition analysis has a few limitations. First of all, since the analysis is all dependent on in-app activities, it can only identify potential churners after they enter the danger zone, such as when they have fewer login times or spend less in the game. However, once the player enters the danger zone, it is sometimes very hard to revert the declining trend as user has already lost interests or encountered bad experience. Also for newly launched games with few in-app activity records available, this kind of attrition analysis will not be helpful to identify potential churners.

If it is possible to identify potential churners by analyzing general trend of app churners outside application, app developers can identify those potential churners at the point of user just entering the app and plan customer retention program targeting at these group of potential churners in the early stage. The research would also be a supplementary to the existing mobile app churn predictions as they generally only consider the in-app activities but not the general characteristics such as users' demographic information.

Since customer behaviors of mobile applications vary from type to type, the following part of the analysis will only focus on Tencent apps. Tencent is a leading provider of media, entertainment, internet and mobile phone value-added services in China, founded in 1998. And in the TalkingData dataset, Tencent is the third app category that contains the most number of apps, with over 49,000 distinct apps. It provides sufficient amount of records for analysis. Also since all these apps are developed by a single company, we may assume that users, as well as churners, share some common characteristics, thus churn prediction can be carried out.

5.3 Hypothesis

Demographic information such as gender and age is related to the attrition possibility.

H1a$_0$: Females are more likely to be churners.
H1b$_0$: Users in younger age are more likely to be churners.
H2$_0$: Users with more apps installed in their devices are more likely to be churners.
H3$_0$: Users who frequently use apps with low correlation with game, such as education apps, are more likely to be churners.
Device technical specifications are related to the attrition probability of a user.
H4a$_0$: Users with smaller device screen are more likely to become churners.
H4b$_0$: Users with smaller internal memory (RAM) more likely to become churners.

By examining these hypotheses using descriptive analysis, variables significant to differentiate between churners and non-churners will be identified and those significant variables will be helpful for app companies to differentiate churners and non-churners.

5.4 Method

5.4.1 Analytical Model

There are four most popular classification methods for churn analysis, namely Naïve Bayes, Logistic Regression, Decision Trees, and SVM. The comparison for these four methods is provided in the table below. ‘

Classification Method	**Advantages**	**Disadvantages**
Naïve Bayes	Short converge time Work well with small dataset	NB conditional independence assumption
Logistic Regression	Probabilistic interpretation Fast and robust result	Must be linearly separable
Decision Trees	Do not need to worry whether the data is linearly separable Ensemble methods like random forests usually have high precision score	Ensemble methods hard to interpret
SVM	Work well with high dimensional data	Memory intensive Hard to interpret

Logistic regression is selected to build the first classifier because of its probability interpretation ability. For example, we can know being a female has positive or negative effect on the user’s churn rate. If the result is not significant, ensemble decision trees and other algorithms will be tried to achieve a higher score.

5.4.2 ETL Process for Variables

The first step of ETL is to filter the data to keep only Tencent app users and their corresponding app events. After finding out all distinct *app_id*s containing category label 'Tencent', all app events having Tencent apps installed (and/or) activated can be retrieved by foreign key *app_id*. All devices that generate these events can, therefore, be correctly identified. After getting the desired *device_id*s, all events belonging to Tencent installed devices were kept in the database and the other records are outside the scope of the analysis. First and last events of each device are also recorded according to timestamp in the table event. After these steps, these outputs are generated to retrieve desired variables.

no. of distinct Tencent app ids: **49,320**
Installed app events: **946,265**
Activated app events: **201,901**
Tencent events: **510,777**
Tencent devices: **40,380**
Tencent user events: **2,231,580**
Tencent user app events: **24,732,155**
Tencent user events (excluding those not inside Tencent user app events): **1,038,496**
Tencent user last event: **40,380**
Tencent user first event: **40,380**

is_churn:

Two kinds of churners are defined with ‘True’ indicating churners.

- [is_churner_uninstalled]: Users who uninstalled all the Tencent apps

If the last event of a Tencent app user is not one of the *Installed app events*, it means the user uninstalled them in the period of 7 days; therefore the user will be marked as a churner

- [is_churner_inactive]: Users who did not use the apps for seven days

If all events of a Tencent app user do not belong to *Activated app events*, the user will be marked as a churner.

installed_app_count:

Number of installed app for the first event of each user was counted.

Device Specifications:

Data regarding device specifications such as screen size, screen resolution and, ram were crawled from ZOL.com. Some data processing processes are carried out to change strings into numeric numbers. The table below presents a summary of results after data processing process. Also some outliers indicating missing data such as 0 and 999 are also removed from the dataset.

VARIABLE	BEFORE	AFTER
SCREEN SIZE	5.5 英寸	5.5
SCREEN RESOLUTION	1920x1080 像素 ☾	1080
RAM	1GB ☾	1000

	512MB ☾	512

Demographics including age and gender are obtained directly from the Kaggle dataset (*gender_age*). Active rate of each app category is the same features used for user profiling and thus same ETL process is applied.

After merging all desired variables on *device_id*, the final training data with 2 dependent variables and 18 independent variables were obtained. The table below shows a summary of available variables and their descriptions.

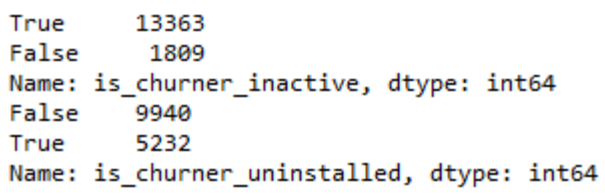
Dependent Variables	*is_churner_uninstalled*	User uninstalled all Tencent apps
	is_churner_inactive	User did not use any Tencent app
Independent Variables	*installed_app_count*	Number of installed app for the first event
	screen_size	Screen size of the device
	screen_resolution	Screen resolution of the device
	ram	Ram of the device
	App active rate	Times of activations for each app category (Data for game category was removed as Tencent apps are classified as game in the app rebelling process)
	is_male	True if the user is male
	age	Age of the user

5.5 Results

5.5.1 Descriptive Analysis on Data

An app category label *game* is actually a subset of *Tencent*, which take up 98.8% of all the Tencent apps. Therefore, the characteristics of Tencent apps might be similar to the mobile games. And this finding is consistent with the app clustering result, that game and Tencent are clustered into a single group marked as the general game category.

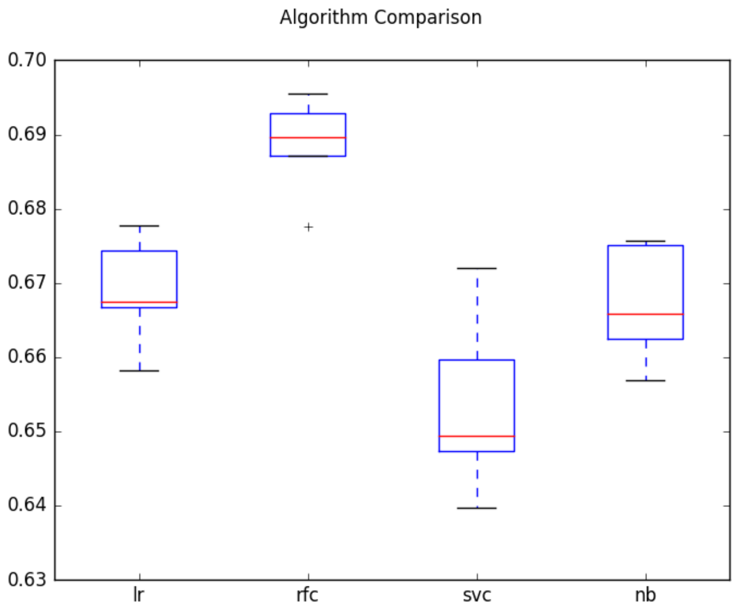
The final dataset contains 15,172 app users, of which 88% are inactive users and 35% uninstalled all Tencent apps. Since majority of the users are inactive users of Tencent apps, which might be able to provide sufficient information to differentiate churners and would cause bias in the classification, I will only choose *is_churner_uninstalled* as the independent variable.



```
True     13363
False     1809
Name: is_churner_inactive, dtype: int64
False    9940
True     5232
Name: is_churner_uninstalled, dtype: int64
```

5.5.2 Model Selection

Four classification models, namely Naïve Bayes, Logistic Regression, Random Forest, and SVM, are applied to train the data with default parameters. 5 folds cross-validation is applied to check the accuracy. From this box plot, Random Forest has the highest prediction score and lowest variance among all, followed by Logistic Regression. Thus the two models are selected for further investigation.



For Logistic Regression, value of parameter *C* (Inverse of regularization strength) was tuned to find the best model.

```
Performing grid search for lr
parameters:
{'C': [0.01, 0.1, 1, 10, 100]}
done in 14.640s
Best score: 0.669
Best parameters set:
        C: 1
```

The accuracy rate after tuning is still very similar to the initial level, showing little improvements.

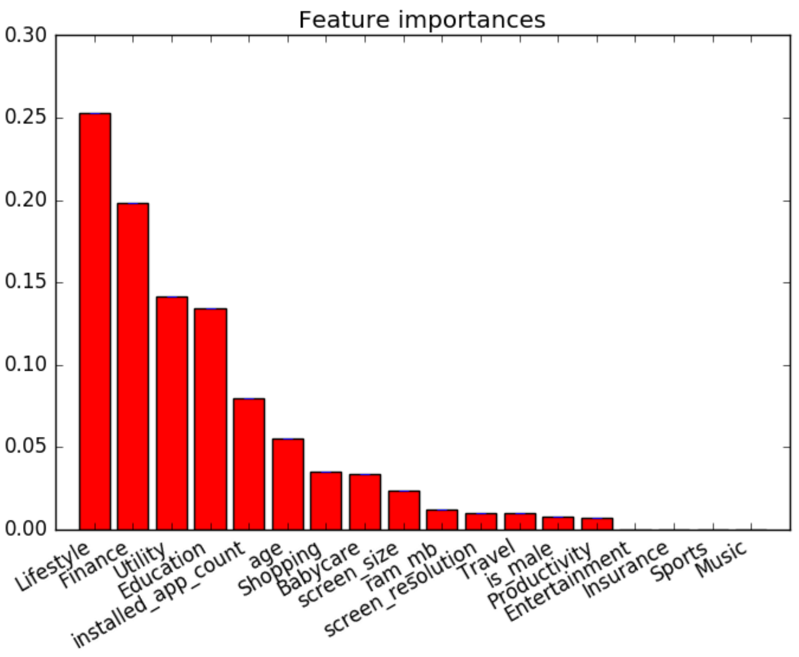
For Random Forest, the following three parameters were altered in the grid search to find out the best model:
[max_features]: The number of features to consider when looking for the best split
[n_estimators]: The number of trees in the forest
[min_samples_leaf]: The minimum number of samples required to be at a leaf node

```
Performing grid search for rfc
parameters:
{'max_features': [None, 'sqrt', 0.2],
 'min_samples_leaf': [1, 20, 50, 80],
 'n_estimators': [5, 10, 50]}
done in 222.433s
Best score: 0.713
Best parameters set:
	max_features: 'sqrt'
	min_samples_leaf: 20
	n_estimators: 50
```

In total, 36 different models are built, reaching a best accuracy score of 71.2%, reflecting a 2.4% increase in accuracy.

5.5.3 Feature Importance

According to the best Random Forest model build, the chart of importance of features is provided below.



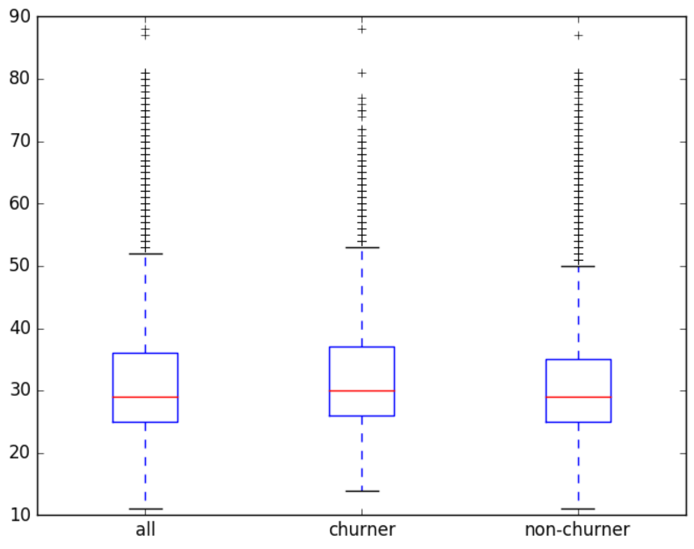
App activation rate for some app categories such as Lifestyle apps and Finance apps have the largest contribution to the model building, followed by no. of apps installed, whereas users' demographic information and device specification have little impacts.

5.5.4 Hypothesis Examination

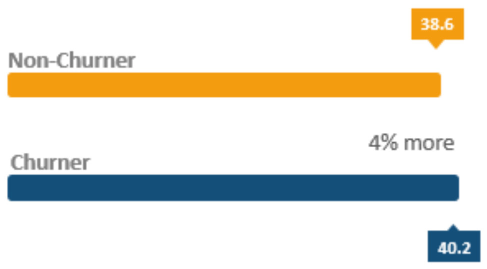
Demographic information such as gender and age is related to the attrition possibility.
H1a$_0$: Females are more likely to be churners.
False. Although it is true that there are more male users than female users, females are actually less likely to become churners. The churn rate for females is 33.2%, which is lower than the male's 35.1%. Also, gender is ranked number 13 among 18 features, which means it is not an important factors to identify churners.

```
is_churner_uninstalled  is_male
False                   0          3486
                        1          6454
True                    0          1734
                        1          3498
dtype: int64
```

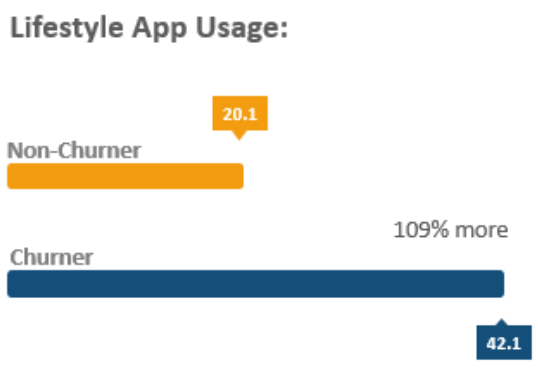
H1b₀: Users in younger age are more likely to be churners.
False. Age of churners are generally higher than non-churners. A possible reason might be people in older age have less time for games, resulting in higher uninstallation rate.

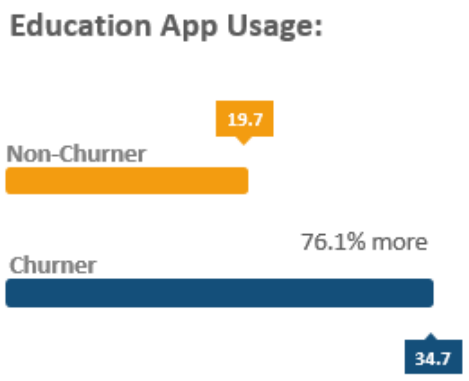


H2₀: Users with more apps installed in their devices are more likely to be churners.
True. Churners generally installed 4% more apps than non-churners. Although the difference is small, the factor is ranked number 5 among 18 features, it might be powerful combining with other factors.

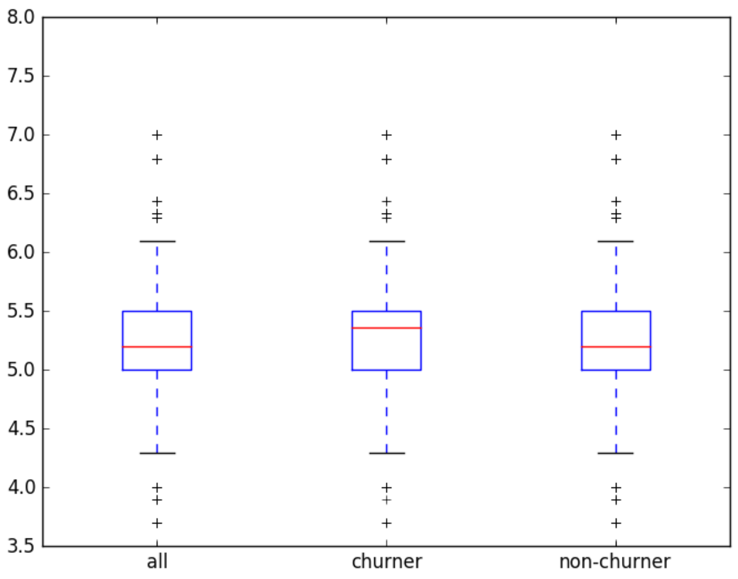


H3₀: Users who frequently use apps with low correlation with game, such as education apps, are more likely to be churners.
True. They are the top factors in churn prediction. For lifestyle apps, churners use them 109% more actively than non-churners. For education apps, they are 76.1% more active. Therefore, users who are more active in these kinds of apps have larger possibility to be churners.

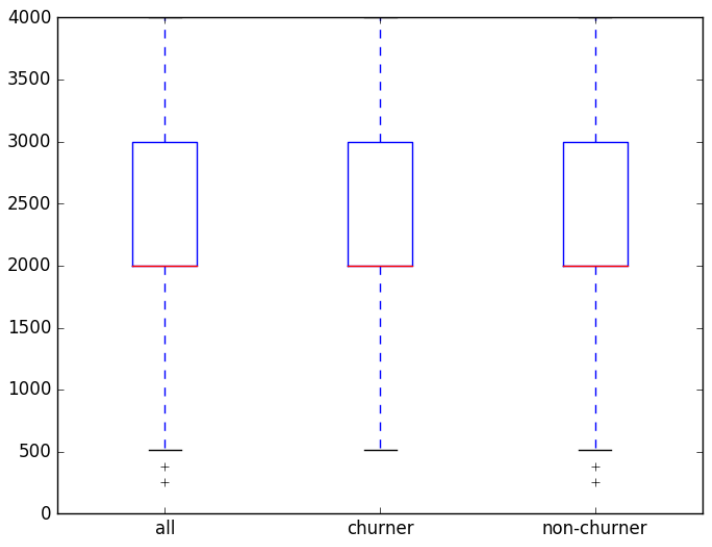




Device technical specifications are related to the attrition probability of a user.
H4a$_0$: Users with smaller device screen are more likely to become churners.
False and it is also insignificant according feature importance.



H4b$_0$: Users with smaller internal memory (RAM) more likely to become churners.
False and it is also insignificant according feature importance



5.6 Conclusion

Churn prediction based on characteristics and activities outside of the application is conducted. Random Forest model reaches a highest accuracy rate of 71.2%, with confusion matrix

$$= \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

$$= \begin{bmatrix} 8689 & 2628 \\ 1251 & 2604 \end{bmatrix}$$

From the confusion matrix, the classifier is observed to have good performance on predict non-churners. However, for those who are predicted to be churners by the classifier, the result is no better than random guess. Since, the cost of engagement activities of mobile apps is much lower than then attrition cost, it is still reasonable for Tencent to target the whole population who are predicted to be churner to prevent future attrition.

Moreover, the churn prediction is able to generate more insights combining with association rule for app recommendation. For example, installation of games will lead to installation of lifestyle apps; however, people who use lifestyle apps more actively have much higher possibility to uninstall Tencent Apps. Therefore, although Tencent could generate higher advertising income by posting lifestyle apps' advertisement, it would suffer from higher attrition rate.

In conclusion, the churn analysis is able to identify potential churners in early stage as it does not rely on any in-app activities. Therefore, it allows early engagement of potential churners to prevent attrition.

Furthermore, it fills in the research gap as it examined a different set of variables from current churn prediction algorithms. Therefore, the significant variables such as app activation rate and number of apps installed can be added to the model to improve the current algorithms.

However, since the data we are using is a sampled dataset, *is_churner_inactive* was not labelled correctly and app activation rates calculated using sampled data cannot fully represent users' app usage. If a complete dataset can be obtained, similar approach could be applied to determine *is_churner_inactive* and more accurate result might be obtained.

6. Installation Rate Prediction

Compared to the iOS and Windows platforms, which have a rather fixed set of device models, the Android platform is adopted by a diverse set of device manufacturers and models. Indeed, the flexibility of the Android platform have greatly contributed to its popularity. To date, the Android platform holds more than 80% of the smartphone market share (Lu et al., 2016). However, such openness has led to a highly fragmented device model set in the market. For app developers, this means extra time and efforts to be spent on different device models for app design, development, maintenance, quality assurance, and revenue generation (Han et al., 2012).

There are existing methods for identifying top device models in each app category, helping developers to improve functionality and usability of apps (Lu et al., 2016). However, the relationship between device-specific factors and user engagement with certain app or certain group of apps has not been investigated.

Thus, in part of this project, we provide a systematic study of device specific factors in relation with app installation status. These factors include in-market price of device model, newness of device model, and hardware specifications including screen size, display (i.e. screen resolution), phone cameras, computation power (i.e. number of core processors), internal storage (i.e. RAM), and battery capacity. We aim to predict the percentage of unique devices installed with targeted app(s), a.k.a, installation rate, for any device model.

With comparison of actual statistics, developers can gain insights of whether the app needs to be further optimized for the device model. If actual app installation rate is far below predicted rate, developers should conduct device-specific app optimization.

6.1 Literature Reviews

We revealed literature on both mobile app installation rate prediction and differences of user-app interaction patterns with different device models.

On mobile app installation rate prediction, the research focus was on individual level (Pan, Aharony & Pentland, 2011). 55 graduates from a US university were recruited as participants in 2010, and their mobile application installation behaviors on a single type of device model were recorded. The paper predicted user installation rate for an app given the user's app adoption status. It used 6 independent variables: call log network, bluetooth network, friendship network, affiliation network, individual variance, and app popularity. Root Mean Squared Error was used for parameter tuning.

On user-app interaction with respect to different device models, it was found that for different device models, there are different traffic volume characteristics for apps, level of user interactions and session lengths, and app usage time (Xu et al., 2011; Patro et al., 2013; Lu et al., 2016).

6.2 Problem Statement

Installation rate for device models refers to the percentage of unique devices installed with targeted app(s). It is calculated by the ratio of the number of devices installed with the app(s) to the number of all devices of a particular device model.

We have 3 hypotheses in Section 6:
Ha_0: App installation rate is correlated with the on-market prices of devices models.
Ha_1: App installation rate is not correlated with the on-market prices of devices models.

Hb_0: App installation rate is correlated with the newness of devices models.
Hb_1: App installation rate is not correlated with the newness of devices models.

Hc_0: App installation rate is correlated with some devices model's technical specifications, such as screen size, processing power.
Hc_1: App installation rate not is correlated with any devices model's technical specifications, such as screen size, processing power.

6.3 Research Methodologies

6.3.1 Data Context

We have 2 data sources, TalkingData dataset for app records on mobile devices, and ZOL dataset for device model specifications. As in our project's context, we consider Tencent apps as the targeted app group for app installation rate investigation, as Tencent is a leading provider of media, entertainment, internet and mobile phone value-added services in China. With over 49,000 distinct apps in this targeted group, we are able to largely avoid sparsity in the dataset selected.

6.3.2 Overview of ETL and Data Scraping Processes

ETL of Data from TalkingData

Although 1667 device models exist in the dataset, some of them do not have labelled app records. Thus, we need to deleting the device models without app records. A MySQL database is firstly created, and inner join operations between relevant tables are conducted. 58,462 unique devices and 1,217 unique device models are identified to be effective, with relevant events and app events. More details can be seen in the comments in the file S0_Data Cleaning_TalkingData.sql.

In addition, app_id with the label of "Tencent" are identified. Devices with events containing these app_id are identified. In total, there are 39,281 unique devices with Tencent apps installed.

Data Scraping from ZOL

As for the device model specifications, the month of first appearance, on-market price as in September 2016, and technical specifications are scraped from the various device model homepages and parameter pages on www.zol.com.cn. The technical specifications include screen size, display (i.e. screen resolution), phone cameras, computation power (i.e. number of core processors), internal storage (i.e. RAM), and battery capacity. Using python package BeautifulSoup, the data scraping follows the following processes:

Step 1: Obtain the list of device models for all brands identified.

Step 2: Obtain the list of device models without brand registered (tablets).

Step 3: Match device models by names on ZOL and TalkingData.

Step 4: Obtain device model specifications from individual product homepage and parameter pages.

With such a process, specifications of 906 unique device models representing 98.4% of all unique devices are found. All major device models (i.e. with number of unique devices greater than 10) are accounted for.

ETL of Data from ZOL

The ZOL dataset for device model specifications is then cleaned and merged with TalkingData dataset for app records on mobile devices with the following processes:

Step 1: Select the most basic device model if duplicated records are registered under the same name. This is largely because the same device model may be customized to the three different operators in China in terms of the internal storage.

Step 2: Transform features selected to ratio data. For instance, 512MB is transformed to 0.5; “单核” is transformed to 1. Cases with missing data are discarded.

Step 3: Merge device models with the same features, so that each device model has a unique set of features. A device model may represent different devices of different brands and names.

Step 4. Merge with Tencent app installation records from TalkingData and compute installation rate for each device model.

After merging device models of same set of specifications, 842 unique device models with Tencent app installation status remain. In the later modelling process, the device models are further merged, to ensure each device model only has a specific set of features. Mean installation rate is 67.3%.

6.3.3 Analytical Models

Model Selection

User-based collaborative filtering and k nearest neighbor (i.e. kNN) are considered. Both method involves using pairwise similarity between the individual cases (i.e. observations). However, collaborative filtering involves the use of all other cases in predicting the targeted case, while kNN only uses the nearest cases.

These methods are selected based on the significant correlation between features (Figure 6.1) may induce a high collinearity. For instance, the correlation between display (i.e. screen resolution) and internal storage (i.e. RAM) is as high as 0.81. This medium-high level of correlation, unlike high correlation such as 0.97, is unable to be eliminated with simply removing any attribute. The factors that have statistically significant impacts on installation rate should be further investigated with running analytical models.

Thus, common regression methods such as Ordinary Least Square Regression with Stepwise Selection may not be suitable. This can also be validated by running a preliminary model on installation rate prediction. The significant features are largely different by running forward and backward selection.

On the other hand, regression with penalization methods such as Ridge Regression and LASSO Regression may be able to deal with collinearity by reallocating the weights of coefficients, they are mainly used to deal with class prediction and not able to predict ratio data in our case.

Although we are able to artificially categorize the installation rates with small bins for accurate prediction, such methods may not be reliable as the cases fall on the class-separation hyperplanes (i.e. the binding cases) may fall into completely different classes with just a small change in the definition of the bin.

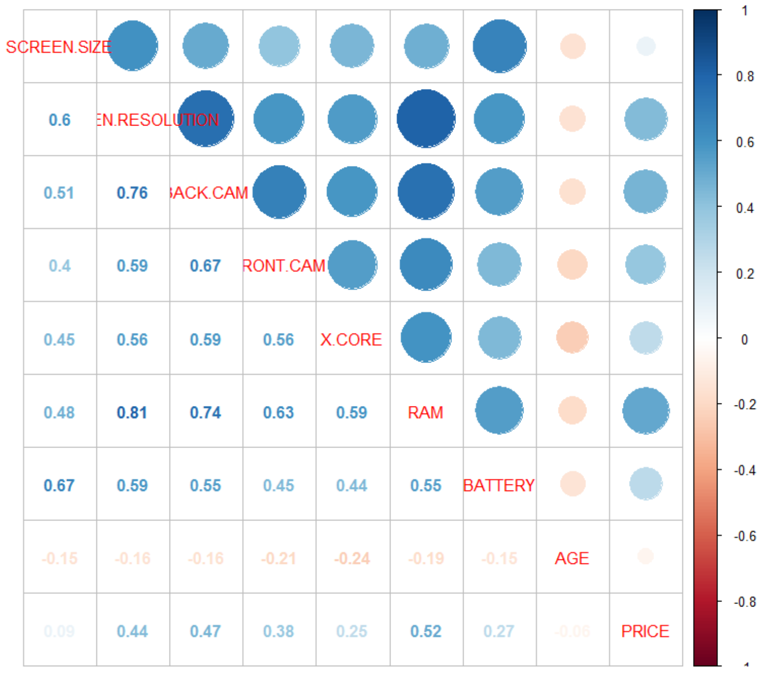


Figure 6.1: Correlation between features

Variables: IV, DV, CV

For the variables, dependent variable is installation rate for app(s) on a targeted device model.

The installation rate is computed as the ratio between the number of unique devices with Tencent apps and the number of unique devices. There are 842 unique device models with installation rates. Root mean squared error (RMSE) is considered as an evaluation measure.

Independent Variables are installation rate for app(s) on other device models, similarity between targeted and other device models based on a set of features selected, and the number of most similar device models used in the prediction (i.e. k in k nearest neighbor method).

As for the pairwise similarity, cosine similarity is used due to three main reasons. Firstly, some attributes have significantly larger mean than the others (i.e. price vs number of core processor) and none of them should be dominating. Thus, scale-variant metrics such as Euclidean distance are not adopted. Secondly, the data is ratio data with a relatively meaningful 0 point. Thus, metrics such as Pearson Correlation Coefficient are not adopted. Thirdly, the dataset is skewed in some attributes such as display, and metrics that assume Gaussian distributions including Mahalanobis distance may not be used.

Since 9 features are identified in total, there are 2^9 = 512 possible combinations of the features. Thus, there are 511 ways of selecting features to for computing similarity between feature vectors. Other than the newness of device model, which is the number of months since the device model's release, and the on-market price of the model, all other attributes are technical features and do not change with time.

Control Variables are the group of apps for installation rate prediction and the time scope of the app installation status on devices. The targeted group is the apps with Tencent label and the time scope is 7 days.

6.4 Results and Discussion

6.4.1 Feasibility of kNN and Collaborative Filtering

The first step is to analyse whether our models are feasible and whether they are able to outperform the baseline method. We consider the predicting installation rate using the mean installation rate as the baseline method. RMSE is used as an evaluation criterion.

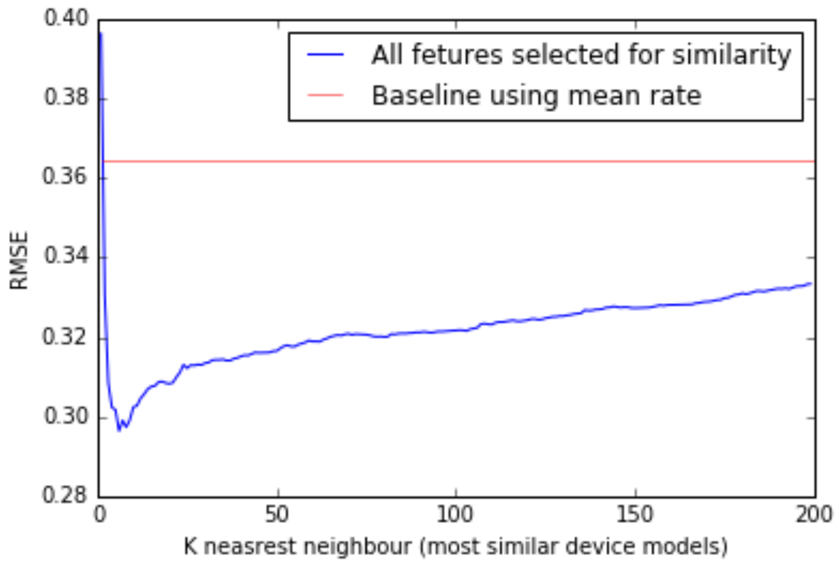


Figure 6.4.1 Initial trial for kNN: with various k selected

In this model, we only vary the number of nearest similar device models selected in prediction. In the other words, the similarity measure is considered as a control variable, with the set of features fixed to be all features. As from the output shown in Figure 6.4.1, we are able to draw 3 conclusions.

1. kNN outperforms the baseline method and is feasible for improving the prediction of installation rate.

2. kNN outperforms collaborative filtering, where by all observations other than the targeted observation is used (i.e. k = n - 1 = 841). This is shown by the gradual and smooth increase of RMSE when k increases.

3. A suitable range of k may be 0 to 30 for later models of different similarity measures. In this range, RMSE initially experience a sharp drop and later experience gradual increase. A range, rather than k* for achieving lowest RMSE in this model, is selected for future models the k* is particularly optimised for dataset with 9 features (i.e. overfitting). If a different set of features are selected, device models with the same set of features are merged together and n may drop.

6.4.2 Device Model Prioritization

Looking into the depth of the algorithm, n(n-1)/2 pair-wise similarity need to be computed. As heavy computational power is spent on this, we may consider reducing the dataset. That is, to leave out unpopular device models and only to predict installation rate for popular device models.

With such an idea, we consider only the device models that are with ***d*** unique devices or more in the dataset. For each reduced dataset, the similarity measure is the cosine similarity between all 9 features and k ranges from 0 to 30. The min RMSE for various k and the baseline RMSE is computed for each ***d***. In addition, the percentage change of RMSE compared to the baseline method is computed.

As seen from the results in Figure 6.4.2, RMSE drops by the largest percentage when the minimum device count is 1136. In this scenario, RMSE is reduced from 0.115 to 0.054. However, only installation rated of the 7 most popular device models are measured. Such significant reduction of data induces a loss of the geniality in our model.

Thus, data reduction by only considering the more significant models is not feasible.

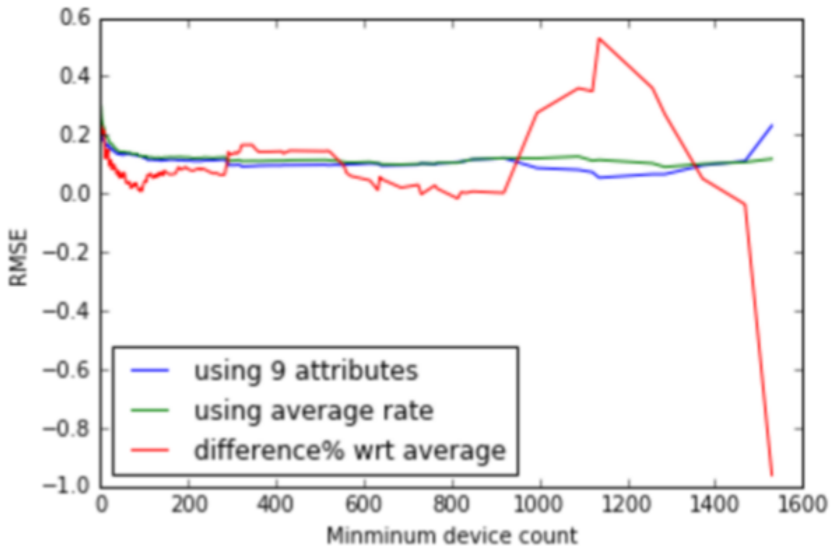


Figure 6.4.2: RMSE for reduced dataset

6.4.3 Feature Selection Models

In the model, the baseline RMSE and min RMSE of selecting different k from 1 to 30, are calculated for all 511 feature subsets. Then, for each possible number of features (1 to 9), significant subset(s) with k* and min RMSE achieved are shown in Table 6.4.3.

A feature set is considered significant, if the min RMSE achieved is the lowest among all feature sets containing the same number of features. The features identified in the significant feature set are in the strong correlation with Tencent app installation rate compared to unselected features, due to the larger min RMSE achieved alternately.

The following observations can be drawn from Table 6.4.3:

1. Using a single feature in predicting the installation rate performs worse than using no feature to predict installation rate. This is illustrated by the -8% reduction in RMSE compared with baseline highlighted in red.

2. Using screen size, front camera resolution, number of core processors and internal storage (i.e. RAM) improves upon the baseline the by 33%. The min RMSE drops to 0.233 when k* is 7 with this feature set. This is the lowest RMSE possible.

3. Price is the most insignificant attribute. Price exists in no significant feature set if not all features are selected. Estimated order of significance: Internal storage (i.e. ram) > Screen size > Front camera resolution > Number of core processor > Display > Rear camera resolution> Newness > Battery capacity > Price

Feature	# Device Models	Baseline RMSE	Min RMSE	Reduction%	k*	Significant Feature Set
1	11	0.248	0.270	-8%	N.A.	#core
2	203	0.331	0.248	33%	18	ram, newness
3	194	0.314	0.235	34%	8	screen size, front cam, ram
4	243	0.309	0.233	33%	7	screen size, front cam, #core, ram
5	318	0.312	0.248	26%	12	all except price, battery, newness, rear cam
6	414	0.331	0.270	23%	10	all except price, battery, newness
7	743	0.358	0.287	25%	14	all except price, battery
7	792	0.361	0.287	26%	9	all except price, #core
7	781	0.358	0.287	25%	8	all except price, screen size
8	798	0.362	0.290	25%	9	all except price
9	842	0.364	0.296	23%	6	all

Table 6.4.3: Model Output: Significant Feature Sets with k* and min RMSE

For the significant feature set with the lowest RMSE, the RMSE can be further improved by device model prioritization. RMSE can be as low as 0.066 when only considering the most popular 13 device models with minimum device count of 940. However, as this only accounts for 36% of all devices, we keep to the full dataset without device model prioritization.

6.4.4 Hypothesis Examination

On-market Price

Ha_0: App installation rate is correlated with the on-market prices of devices models.
Ha_1: App installation rate is not correlated with the on-market prices of devices models.

Ha_0 is indeed false, with the observation that price is the most insignificant feature among all 9 features.

One possible explanation is that price is time variant. While new models' on-market prices are set by retailers and relatively accurate in representing consumer's willingness to purchase, old models' on market prices are not. The prices can either be the retail price set long time ago and are much higher than current consumer's willingness to purchase, or be estimated values in second-hand market which are subjected to human errors.

Newness

Hb_0: App installation rate is correlated with the newness of devices models.
Hb_1: App installation rate is not correlated with the newness of devices models.

Hb_0 is false to a large extent. Newness is the 3rd least significant factor which does not exist in most of the significant feature sets. In addition, RMSE improves from 0.287 to 0.270 when newness is removed.

Technical Specifications

Hc_0: App installation rate is correlated with some devices model's technical specifications, such as screen size, processing power.
Hc_1: App installation rate not is correlated with any devices model's technical specifications, such as screen size, processing power.

Hc_0 is true, as the most significant feature set contains technical specifications including internal storage (i.e. ram), screen size, front camera resolution, and number of core processor. This is consistent with existing literature.

6.5 Conclusion

In this section of the project, we conducted a systematic study of device specific factors in relation with app installation status, which filled a current research gap.These factors include in-market price of device model, newness of device model, and hardware specifications including screen size, display (i.e. screen resolution), phone cameras, computation power (i.e. number of core processors), internal storage (i.e. RAM).

We find that app installation rate is indeed correlated with the technical specifications of a device model, including features such as screen size, computation power and internal storage. In addition, we are able to use kNN to predict installation rate more accurate than simply using the mean installation rate, with an improvement of 33% of the RMSE.

With comparison of actual statistics, developers can gain insights of whether the app needs to be further optimized for the device model. If actual app installation rate is far below predicted rate, developers should conduct device-specific app optimization.

In the future, we may conduct panel analysis of the installation rate to investigate the time-invariant effects of device specifications, and reduce our current limitation of overfitting to the particular 7-day data. In addition, we may also investigate installation rate of specific app instead of the app groups.

7 Conclusion

In the report, four kinds of analysis based on TalkingData are conducted.

For user profiling, users' demographic information and active rates of 13 mobile app categories could be used to segment smartphone users. By applying KMeans function of Python scikit-learn library, elements within each resulted cluster are closely packed together. Moreover, by using visualization tool, we can conclude that there is statistical difference in distribution of some app category activity among each cluster.

For app recommendation, by applying Apriori algorithm, some certain correlations between different app categories can be discovered by association rule mining. The results discovered in this section could have a great impact on marketing strategies. Advertisers could maximise the effectiveness of their advertisement campaign by displaying inter-app ads based on our findings.

For attrition analysis, the presented classifiers are applied to analyse how external factors, which are users' demographic information, number of apps installed, apps' activation rate, and device specifications, contribute in predicting churners. Random forest classifier reaches the highest accuracy rate of 71.2%, suggesting apps' activation rate and number of apps installed as key factors.

For installation rate prediction, we have presented a novel kNN model in predicting installation rate more accurate than simply using the mean installation rate, with an improvement of 33% of the RMSE. We find that app installation rate is indeed correlated with the technical specifications of a device model, including features such as screen size, computation power and internal storage.

8 References

Abbas, O. A. (2008, July). Comparisons between Data Clustering Algorithms. *The International Arab Journal of Information Technology, 5*(3), 320-325.

Buettner, R., (2016). "Predicting user behavior in electronic markets based on personality-mining in large online social networks: A personality-based product recommender framework". *Electronic Markets: The International Journal on Networked Business.* Springer: 1–19. doi:10.1007/s12525-016-0228-z.

Coussement, K., & Poel, D. V. (2009). Improving customer attrition prediction by integrating emotions from client/company interaction emails and evaluating multiple classifiers. *Expert Systems with Applications, 36*(3), 6127-6134. doi:10.1016/j.eswa.2008.07.021

Francesco Ricci, Lior Rokach and Bracha Shapira. (2011). *Introduction to Recommender Systems Handbook*. Retrieved from http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf

Fruchterman, T. M., & Reingold, E. M. (1991). Graph drawing by force-directed placement. Software: Practice and experience, 21(11), 1129-1164.

Hadiji, F., Sifa, R., Drachen, A., Thurau, C., Kersting, K., & Bauckhage, C. (2014). Predicting player churn in the wild. *2014 IEEE Conference on Computational Intelligence and Games*. doi:10.1109/cig.2014.6932876

Hamka, F. (2012). *Smartphone's Customer Segmentation and Targeting.* unpublished manuscript, Faculty of Technology, Policy and Management, Delft University of Technology, Netherlands

Han, D., Zhang, C., Fan, X., Hindle, A., Wong, K., & Stroulia, E. (2012).*Understanding Android Fragmentation with Topic Analysis of Vendor-Specific Bugs*. Paper presented at the Proceedings of the 2012 19th Working Conference on Reverse Engineering.

Hung, S., Yen, D. C., & Wang, H. (2006). Applying data mining to telecom churn management. *Expert Systems with Applications, 31*(3), 515-524. doi:10.1016/j.eswa.2005.09.080

Infographic: What Data-Driven Marketing Looks Like in 2015. (n.d.). Retrieved September 16, 2016, from http://www.adweek.com/news/technology/infographic-what-data-driven-marketing-looks-2015-163607

iTunes. (2016). Retrieved from https://itunes.apple.com/sg
Jourdan, A. (2016, June 28). China tightens rules for mobile app developers. Retrieved September 10, 2016.

Koren, Y., & Bell, R. (2010). Advances in Collaborative Filtering. *Recommender Systems Handbook,* 145-186. doi:10.1007/978-0-387-85820-3_5
Kotler, Philip. (2003). *Marketing Management*. New Jersey: Pearson Education.

Li, H., Lu, X., Liu, X., Xie, T., Bian, K., Lin, F. X., . . . Feng, F. (2015).*Characterizing Smartphone Usage Patterns from Millions of Android Users*. Paper presented at the Proceedings of the 2015 ACM Conference on Internet Measurement Conference, Tokyo, Japan.

Localytics. (n.d.). Mobile Apps: What's A Good Retention Rate? Retrieved September 16, 2016, from http://info.localytics.com/blog/mobile-apps-whats-a-good-retention-rate

Lu, X., Liu, X., Li, H., Xie, T., Mei, Q., Hao, D., . . . Feng, F. (2016). *PRADA: prioritizing android devices for apps by mining large-scale usage data*. Paper presented at the Proceedings of the 38th International Conference on Software Engineering, Austin, Texas.

Mobile Internet Trends Mary Meeker 2015 2 - Smart Insights. (n.d.). Retrieved September 16, 2016, from http://www.smartinsights.com/?attachment_id=53812

Patro, A., Rayanchu, S., Griepentrog, M., Ma, Y., & Banerjee, S. (2013).*Capturing mobile experience in the wild: a tale of two apps*. Paper presented at the Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, Santa Barbara, California, USA.

Pan, W., Aharony, N., & Pentland, A. S. (2011). *Composite social network for predicting mobile apps installation*. Paper presented at the Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, California.

Poushter, J. (2016, February 22). Smartphone ownership rates skyrocket in many emerging economies, but digital divide remains. Retrieved September 10, 2016.

Reps J., Aickelin U. & Garibaldi J., 2015, Personalising Mobile Advertising Based on Users' Installed Apps. Retrieved Sep 10, 2016.

Vijayalaksm, S. (2012, December). A Fast Approach to Clustering Datasets using DBSCAN and Pruning Algorithms. *International Journal of Computer Applications, 60*(14).

Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J., & Venkataraman, S. (2011). *Identifying diverse usage behaviors of smartphone apps*. Paper presented at the Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, Berlin, Germany.

Xia, X., Wang, X., & Zhou, X. (2013) . *Evolving Recommender System for Mobile Apps: A Diversity Measurement Approach.* Smart Computing Review, vol. 3, no. 3, June 2013

Zhongguancun Online. (2016). Retrieved from http://www.zol.com.cn.