American University of Armenia

Akian College of Science and Engineering

Capstone Project

Mobile Application User Segmentation and Churn Prediction

Liana Mehrabyan
Supervised by: Arnak Dalalyan

Spring, 2018

Abstract

In the era of smart phones, mobile application development has become one of the leading fields in business and keeping the users engaged along with forecasting their future behavior are one of the main challenges in this field. This project focuses on mobile application users' segmentation and churn prediction based on 95.000 users' app usage data. The methods used to segment the users vary from marketing techniques as RFM analysis to Gaussian Mixture Models. Considering that knowing whether the user deleted the application or not is impossible to retrieve, a definition of user churn is suggested and used to label the data. Finally, binary classification models are built for the labeled data in order to predict customer churn.

Keywords: Mobile Application Development, Unsupervised Learning, Gaussian Mixtures, Expectation Maximization Algorithm, RFM Analysis, Renewal Processes, SMOTE, Clustering, Supervised Learning, Classification.

Contents

Abstract **i**

1 Problem Statement and Motivation **2**
1.1 Motivation Behind App User Segmentation 2
1.2 The Data 3

2 Proposed Approaches and Theoretical Background **5**
2.1 User Segmentation 5
2.1.1 RFM Customer Analysis 5
2.1.2 Gaussian Mixture Models 6
2.1.3 Expectation Maximization Algorithm 7
2.1.4 Decision Trees 9
2.1.5 Random Forest 10
2.2 Data Labeling 10
2.2.1 Alternating Renewal Processes 10
2.3 Churn Prediction 11
2.3.1 Classification Techniques 11
2.3.2 SMOTE Oversampling Technique 14

3 Experimental Results **15**
3.1 Preliminary Analysis and User Segmentation 15
3.2 Definition and Prediction of Customer Churn 19
3.2.1 Conclusion 23

Chapter 1

Problem Statement and Motivation

1.1 Motivation Behind App User Segmentation

In the age of technological advancement and heavy usage of mobile phones, mobile application development has become one of the leading business fields. In 2015, the global mobile app revenues resulted in 69.7 billion US dollars while 2020 revenue predictions reach 188.9 billion US dollars. The main sources of an app's revenue are app store purchases, in-app purchases and in-app advertisement. Therefore, having high download rates is not a criterion of a successful app and consistent user activity is the main value of interest. In average, 21% of users use the app just once and it is crucial to understand what leads to user churn.[1]

The ability of tracking users' behaviour while using the app results in very insightful data which, if being well analyzed, can considerably improve app retention rate. Analysis of such data can lead to highlighting some app drawbacks that cannot otherwise be seen, segmentation of users that will help to make the app more personalized and user-satisfying as well as identifying users that are at risk of churning. The latter can be an important tool that will help to save that users by re-marketing techniques, sending push notifications or making the app more individualized. While a mobile app developer can retrieve the number of users at the beginning and the end of a time period, in practice, it is impossible to retrieve whether a certain user has deleted the app or not. Hence, it is important to have a definition of app churn that will be statistically meaningful and have a profound theoretical basis.

Given data of a real estate mobile app users' behavior, this project focuses on identification of different user subgroups and definition of user churn. It also includes methods of supervised learning that have been developed in order to identify users that are prone to attrition.

1.2 The Data

The initial data set provided for this project represents a real estate application usage records of around 90.000 users in a six-month time scope. This per-session data set consists of 11.000.000 observations where each observation represents the information about one application usage session of one user. In particular, the data set includes the following features:

- **device_id**: a unique character string identifier for each user.
- **timestamp**: the time stamp of session start represented in UNIX (also known as POSIX) time system where each number represents the number of seconds since the Coordinated Universal Time (UTC), Thursday, 1 January 1970.
- **crashed**: whether or not the session ended as a result of a crash.
- **duration**: duration of the session defined as the time (in seconds) between the session start and termination, including the situations of the app being sent into phone background because of external interactions such as calls or app switches.
- **screens count**: number of app screens generated during the session. (i.g. Log in page, search page, settings page etc.)
- as well as the OS version, app version and country

The data set contained many erroneous entries such as sessions having duration of more than 20 hours or hundreds of screens generated in seconds that have been eliminated from the data set after confirming that these were resulted from an sdk bug. Moreover, the proportion of such observations was not considerable in comparison with the total amount of observations. Considering the user-focused approach of this project, another — user-based dataset with 90.000 observations of the following derived features:

- **device_id, last_session, total duration, average duration, average number of generated screens, average_IAT: average time between two sessions, number of crashes, number of sessions**
- **crash_rate:** the ratio of number of crashes and total
- **max_IAT:** longest time between two consecutive sessions the user ever had

- **R:** "recency" of the user, i.e. the difference between the last time record in the data set and the last session. In practice, the last time record would be considered to be the current date. number of sessions

- **R_score, F_score, M_score and RFM**: numbers derived as a result of RFM customer segmentation analysis that will be discussed in Chapter 2.

Chapter 2

Proposed Approaches and Theoretical Background

2.1 User Segmentation

2.1.1 RFM Customer Analysis

Having user segmentation as one of the main objectives, obtaining features that will have a good user descriptive value in terms of loyalty and overall activity is important as those can have high contribution in the clustering process. One of such measures is the RFM score obtained from RFM customer value analysis method which distinguishes important customers from large data set based on the following three criteria:

- **Recency:** the time between the present and the last product consumption. The higher the interval, the lower is the recency score.
- **Frequency:** number of times the customer uses the product over a certain time interval. The higher the number of usage, the higher the F score.
- **Monetary:** the monetary value of the consumption.

In the context of the project, **Recency** represents the time interval between the last recorded session time in the dataframe and the last session of the user, **Frequency** represents the number of sessions per month and **Monetary** represents the total duration spent using the application, as, in this case, the time is the main value of interest. After having having the R, F and M values, they are being scored on a certain scale from 1 to k, k being chosen according to the nature of the analysis. In this case, a scale of 1 to 3 was used based on what quantile interval the value falls into. Finally, an RFM score is generated from these three values as a three-digit combination of these numbers: $100 * R + 10 * F + M$. [2] [3]

2.1.2 Gaussian Mixture Models

A Gaussian mixture model (GMM) is a probabilistic model that attempts to find normally distributed sub-populations within the overall population, i.e. it finds a mixture of multi-dimensional Gaussian probability distributions that best describe the input dataset. More formally:

Definition 1. *A K-component Gaussian mixture is a weighted sum of K Gaussian densities given by the form:*

$$p(x) = \sum_{k=0}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{2.1}$$

*where each Gaussian density $N(x|\mu_k, \Sigma_k)$ is called a **component** having its mean μ_k and covariance Σ_k and the parameters π_k are called **mixing coefficients**.*

Moreover, the mixing coefficients should satisfy:

$$\sum_{k=0}^{K} \pi_k = 1 \tag{2.2}$$

Combining the constraints of $p(x) > 0$, $N(x|\mu_k, \Sigma_k) \geq 0$ yield $\pi_k \geq 0$ for all k and (2.2) yields $0 \leq \pi_k \leq 1$. It is now verified that π_k-s are valid probabilities.

Define a random variable $\boldsymbol{z}$ of K dimensions such that $z_k = 1$ and all other elements are 0 if x belongs to component k. The probability $P(z_k = 1)$ can be expressed in terms of the mixing coefficient π_k:

$$P(z_k = 1) = \pi_k$$

A complete Gaussian mixture model is described by the parameters $\boldsymbol{\pi}$ =[π_1,π_2, .., π_K], $\boldsymbol{\mu}$=[μ_1, μ_2, ... , μ_K] and $\boldsymbol{\Sigma}$=[Σ_1,Σ_2,...,Σ_K]. Figure 2.1 represents an example of a 4-component Gaussian mixture.

Recall that for a D-dimensional vector x, the mixture component p_k will have the following distribution: [4]

$$p_k(x) = \frac{1}{2\pi^{\frac{D}{2}} |\Sigma_k|^{\frac{k}{2}}} exp\Big\{ - \frac{(x-\mu_k)^T(\Sigma_k^{-1}(x-\mu_k)}{2} \Big\} \tag{2.3}$$

Gaussian mixture models are widely used in data mining, pattern recognition, machine learning, and statistical analysis. In the scope of this project, the contribution of Gaussian Mixture Models is crucial as it identifies underlying user subgroups from the overall user population. The latter will have its detailed discussion in Chapter 3.

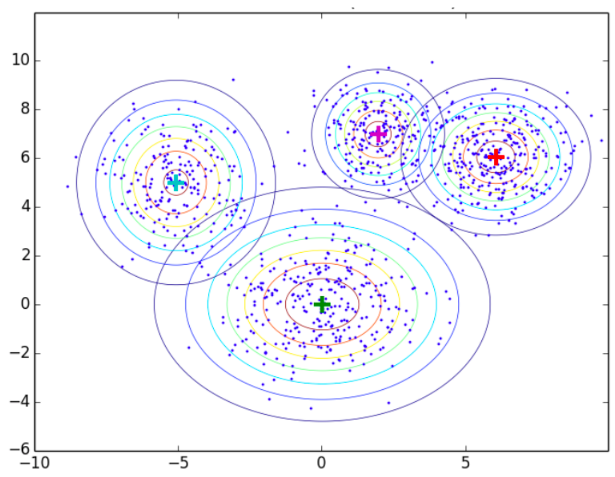

Figure 2.1: Gaussian Mixture with 4 components

Having the approach of Gaussian Mixture Models in mind, the next step is the parameter estimation. In frequentist probability theory, models are typically learned by using maximum likelihood estimation techniques, which maximize the probability or likelihood of the observed data given the model parameters. However, finding the maximum likelihood solution for mixture models is usually analytically impossible. That is why numerical techniques are used to estimate the maximum likelihood. One of the approaches for parameter estimation of GMM is the Expectation Maximization (EM) algorithm that will be discussed in the next section of this chapter.

2.1.3 Expectation Maximization Algorithm

Expectation Maximization algorithm is an iterative algorithm consisting of two main—Expectation (E) and Maximization (M) steps. Given a set of observations x_1,x_2,....x_N to model Gaussian Mixtures, one can represent this data as an NxD matrix $\boldsymbol{X}$ where the i^{th} row is the transpose of x_i. The goal is to maximize the log-likelihood given by:

$$log\ p(\boldsymbol{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln\Big\{ \sum_{k=1}^{K} \pi_k N(x_b; \mu_k, \Sigma_k) \Big\} \tag{2.4}$$

The EM algorithm can be summarized as follows:

1. Initialize the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ and the initial value of the log likelihood.

2. E-step: estimate the posterior probability $\gamma(z_{nk})$ of the i^{th} observation belong-

ing to the k^{th} component:

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n; \mu_k, \Sigma_k)}{\sum_{i=1}^{K} \pi_i N(x_n; \mu_i, \Sigma_i)} \tag{2.5}$$

3. M-step: update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ given the current posterior probabilities:

$$\mu_k^{t+1} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \tag{2.6}$$

$$\Sigma_k^{t+1} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{t+1})(x_n - \mu_k^{t+1})^T \tag{2.7}$$

$$\pi_k^{t+1} = \frac{N_k}{N} \tag{2.8}$$

   where $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$

4. Evaluate the log likelihood

$$log\ p(\boldsymbol{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln\Big\{ \sum_{k=1}^{K} \pi_k N(x_b; \mu_k, \Sigma_k) \Big\} \tag{2.9}$$

5. Check for convergence of the parameters or the likelihood.

There are several considerations that should be handled while opting for the EM algorithm: setting the number of components, a good initial estimate for the parameters and parameterization of the covariance matrix.

In the context of this project, the Bayesian information criterion (BIC) is will be used for choosing the optimal number of components:

Definition 2. *Given a finite set of models, let MLL_i be the maximum log likelihood of the i^{th} model. And let d_i be the dimension of the i^{th} model. Then, the penalty BIC_i for the model M_i is given by:*

$$BIC_i = MLL_i - \frac{1}{2} d_i logn \tag{2.10}$$

[5]

Naturally, the model with the lowest BIC is preferred.

The two most common ways of initializing parameters are either using K-means algorithm or doing a random initialization.

As for covariance matrices, the Expectation Maximization algorithm can consider covariance matrices of three families, namely:

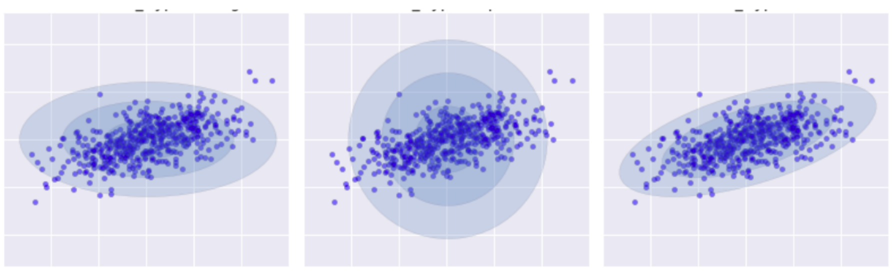

Figure 2.2: Gaussians with covariance matrices from diagonal, spherical and general families.

- Spherical: where each variable of the component density has the same variance so that the distribution is spherical.
- Diagonal: which results in axis-aligned elliptical components because the variance in each dimension is allowed to vary.
- General: in which the covariance matrices are not constrained to be diagonal.

Visual representation of corresponding distribution shapes can be seen in Figure 2.2. An additional type of assumption can be considered while using the EM algorithm, which is the assumption of all components having the same covariance matrix not constrained to be diagonal. Such constrain is called to be tied. [6]

2.1.4 Decision Trees

Decision Trees (DTs) are a non-parametric supervised learning method that takes an input vector of feature values and outputs a "decision" in form of a single value. Both input and output values can be discrete or continuous. However, in the scope of this project, binary outcome values will only be considered, i.e. binary classification DTs. Given observations belonging to two classes, the goal of DTs is to classify those observations. The main idea behind DTs is producing the output value after performing a sequence of splits based on certain rules. The approach for choosing the optimal split is greedy, i.e. the algorithm chooses the split that results in the biggest homogeneity (having observations of the same class) of the splitted subsets.

Definition 3. *Let p_A and p_B be the proportions of observations belonging to class A and B correspondingly in set D. Then,*

$$H(D) = -p_A log_2(p_A) - p_B log_2(p_B) \tag{2.11}$$

is called the entropy of the set D.

Entropy measures the purity or homogeneity of the set and takes its minimum value $H(D) = 0$ when the set is completely homogeneous and $H(D) = 0$ when $p_A = p_B = 0.5$ [7]

2.1.5 Random Forest

Random Forests are an ensamble learning model for classification and regression problems. Again, in the scope this project, binary classifier Random Forests will be considered. RF consist of multiple decision trees and classify given observation based on the majority vote of the decision trees.

Definition 4. *Let p_A and p_B be the proportions of observations belonging to class A and B correspondingly in set D. Then, $I_G(D) = 1 - p_A^2 - p_B^2$ is called the Gini impurity.*

Apart from classification and regression problems, Random forests are a useful tool for evaluating feature importance in a data frame. In particular, given a feature f_i, RF evaluates the decrease in accuracy and its effect on the Gini index in case of excluding f_i from the model. [7]

2.2 Data Labeling

2.2.1 Alternating Renewal Processes

Before proceeding to Alternating Renewal Processes, one should recall the notion of a Stochastic Process. A stochastic process is a process that develops in time according to probabilistic rules. Formally:

Definition 5. *Suppose that (Ω, F, P) is a probability space and $I \subset \mathbb{R}$ has finite cardinality. Suppose further that for each $\alpha \in I$, there is a random variable $X_\alpha : \Omega \to \mathbb{R}$ defined on (Ω, F, P). The function $X : Ix\Omega \to \mathbb{R}$ defined by $X(\alpha, \omega) = X_\alpha(\omega)$ is called a stochastic process with indexing set I, and is written $\{X_\alpha, \alpha \in I\}$. [8]*

A particular type of stochastic processes are the renewal processes in which the events happen randomly in time, i.e. the inter-arrival times of events are iid. Such assumption makes renewal processes very useful for building more realistic models. A particular sub type of renewal processes —Alternating Renewal Processes will be considered in the context of this project.

An alternating renewal process alternates between two states “up" and “down". Imagine a system that is first up for time U_1 then down for D_1 then it is up for time U_2 and so on. Define $\{U_n, n \geq 1\}$ and times system being down $\{D_n, n \geq 1\}$.

Consider a state variable $Z(t)$ that is 1 is the system is up at time t and 0 if the system is down at time t.

Definition 6. *A renewal process $N(t), t \in T$ with a state variable $Z(t)$ and duration sequences D_n and U_n is called an alternating renewal process.*

It is required that the sequences of pairs $\{(U_n, D_n) : n \geq 1\}$ be i.i.d. random vectors.

Theorem 1 (Renewal Reward Theorem). *For a positive recurrent renewal process in which a reward R_j is earned during cycle length X_j and such that $\{(X_j, R_j) : j \geq 1\}$ is iid with $E[R_j] < \infty$ the long run rate at which rewards are earned is given by*

$$\lim_{x \to \infty} \frac{R(t)}{t} = \frac{E[R]}{E[X]} \tag{2.12}$$

where (X, R) denotes a typical "cycle" (X_j, R_j).

One of the main properties of Alternating Renewal processes that is derived from Renewal Reward Theorem is the ability to deduce the long-run proportion of time that the alternating renewal process is up or down:

$$\text{long-run proportion up} = \lim_{x \to \infty} \tfrac{1}{t} \int_0^t Z(s)ds = \tfrac{E[U]}{E[U]+E[D]}$$

$$\lim_{x \to \infty} P(Z(t) = 1) = \tfrac{E[U]}{E[U]+E[D]}$$
$$\lim_{x \to \infty} P(Z(t) = 0) = \tfrac{E[D]}{E[U]+E[D]}$$

A more detailed introduction to Renewal Processes and the proof of the Renewal Reward Theorem can be found in [9] and [10]. The application of these properties will be seen in the further chapters of this paper in order to define user churn.

2.3 Churn Prediction

2.3.1 Classification Techniques

Naive Bayes Classifier is a generative probabilistic model that assumes mutual independence between the features and classifies given observation based on Bayes' rule. Recall,

Theorem 2 (Bayes' Rule). $p(y|x) = \frac{p(x,y)}{p(x)}$

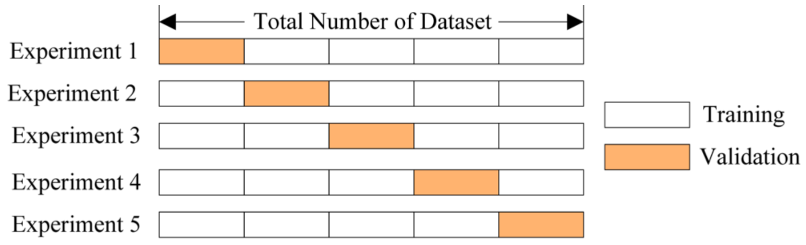

Figure 2.3: Cross Validation Technique

Given a classification problem with classes $C = \{0, 1\}$ and set $\{\phi_1, \phi_2, ..., \phi_k\}$ of normally distributed features, Naive Bayes classifiers assumes that:

$$P(c|\phi_1, \phi_2, ...\phi_n) \propto P(c) \prod_{i=1}^{n} P(\phi_i|c)$$

Where $P(c|\phi_1, \phi_2, ...\phi_n)$ is the posterior probability of class, $P(c)$ is the prior probability of class. The class resulting in a higher probability is then assigned to the observation under consideration.

K Nearest Neighbors is a similarity based model that identifies K nearest neighbors to the observation that has to be classified and assigns the class based on the majority vote among those K neighbors. K nearest neighbors are be chosen by calculating distances using different distance metrics as Euclidean, Manhattan, Minkowski etc., based on the nature of the problem or the characteristics of the features. Choosing the optimal number of K neighbors is usually done by cross-validation technique the idea of which is summarized in Figure 2.2. Different numbers of K are tested in a given range using cross-validation and the optimal number of K is chosen for the final model. [7]

Support Vector Machine is a max-margin classifier the output of which is an optimal hyperplane which classifies new examples. Before proceeding to details, the notion of linearly separable set should be recalled. Without formal definition, a set D is called linearly separable if there is a hyperplane that perfectly separates the examples in D. Given a linearly separable training set D, a max-margin classifier maximizes the distance between the observations in D and the decision boundary which is called the margin. Such decision boundary corresponds to the observations $x \in D$ such that $\omega_0 + \boldsymbol{\omega^T} \phi(\boldsymbol{x}) = 0$, where $\boldsymbol{\omega}$ is a vector of real-valued weights $[\omega_1 \ \ \omega_2 ... \ \omega_k]^T$ and $\boldsymbol{\phi(x)}$ is the feature set. States that fall on the two sides of decision boundary satisfy one of these conditions: $\omega_0 + \boldsymbol{\omega^T} \phi > 0$ or $\omega_0 + \boldsymbol{\omega^T} \phi < 0$. Considering all these, the following optimization problem is derived:

Lemma 1. *Given set of classes a_n, the max-margin classifier is the solution to the constrained optimization problem:*

$$minimize \frac{1}{2}||\boldsymbol{\omega}||^2 \tag{2.13}$$

$$subject\ to\ a_n(\omega_0 + \boldsymbol{\omega^T} \boldsymbol{\phi(x_n)}) \geq 1, n = 1, ..., N.$$

By writing the Lagrangian:

$$L(\boldsymbol{\omega}, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\omega^T\omega} + \sum_{n=1}^{N} \alpha_n(1 - \alpha_n(\omega_0 + \boldsymbol{\omega^T} \boldsymbol{\phi(x_n)})) \tag{2.14}$$

and corresponding KKT conditions:

$$\boldsymbol{\omega} = \textstyle\sum_{n=1}^{N} \alpha_n a_n \phi(x_n)$$
$$\text{and}$$
$$\textstyle\sum_{n=1}^{N} \alpha_n a_n = 0$$

Plugging this back to the Lagrangian and re-writing the KKT conditions yields:

$$\alpha_n \geq 0$$
$$\alpha_n \omega_0 + \boldsymbol{\omega^T} \boldsymbol{\phi(x_n)} - 1 \geq 0$$
$$\alpha_n(1 - a_n(\omega_0 + \boldsymbol{\omega^T} \boldsymbol{\phi(x_n)})) = 0$$

This concludes that either $\alpha_n = 0$ or $a_n(\omega_0 + \boldsymbol{\omega^T} \boldsymbol{\phi(x_n)})) = 1$ meaning that the multipliers α_n are all zero except for those examples falling in the margin. These are called support vectors identification of which is the core of max-margin computation.

In practice, however, linearly separable training sets are hardly met. To extend the application of SVM, an term is added to (2.13) that penalizes the observations falling into the margin. The resulting problem is the following:

$$minimize \frac{1}{2}||\omega||^2 + C \sum_{n=1}^{N} \varepsilon_n \tag{2.15}$$

$$\text{subject to } a_n(\omega_0 + \omega^T \phi(x_n)) \geq 1 - \varepsilon_n, n = 1, ..., N.$$

The constant C regularizes a trade-off between the width of the margin and how much "violations" to the margin are accepted. The SVM classifier is determined by solving the associated optimization problem presented in (2.14). The solution, however, is left outside of the scope of this project. It should be noted that the solution methods include gradient-based approaches or specialized methods for quadratic programming.

2.3.2 SMOTE Oversampling Technique

Synthetic Minority Over-sampling Technique (SMOTE) is an approach of creating synthetic observations in order to handle the imbalance problem in a dataset. A dataset is called imbalanced if there is a considerable difference between the proportion of classes in the target variable. Performance of many machine learning algorithms is highly affected by class imbalance and it is important to give a sensible solution to this problem meanwhile avoiding possible overfitting. One of the approaches is resampling the original data, either by oversampling the minority class and/or undersampling the majority class, where oversampling is done with replacement. This, however, does not improve performance considerable and also can create bias. What distinguishes SMOTE is its approach of creating synthetic observations. Depending on the amount of oversampling that is required, neighbors from K nearest neighbors of a minority class observation are chosen. The difference between the observation under consideration and the neighbor is calculated, multiplied by a random number between 0 and 1 and added to the observation. [11] This causes the selection of a random point along the line segment between two specific features. Such approach makes the decision region of the minority class to become more general yet avoiding duplicates in the data.

Chapter 3

Experimental Results

3.1 Preliminary Analysis and User Segmentation

At the outlier detection stage of the preliminary analysis of the users' data set, a hypothesis of having mixture data emerged and it was concluded that the classical approaches for univariate outiler detection such as boxplot visualizations are not appropriate for this case unless different user groups are identified in the data. Figure 3 represents the initial boxplots of duration and number of screens that approve the possibility of having mixtures. Moreover, already in the initial stage of data exploration in the session-based data a distinction of several, yet unknown, different types of sessions could be sensed which is assumed to be generated from different user groups. Before applying the algorithm, however, several outliers have been removed using visualization techniques.

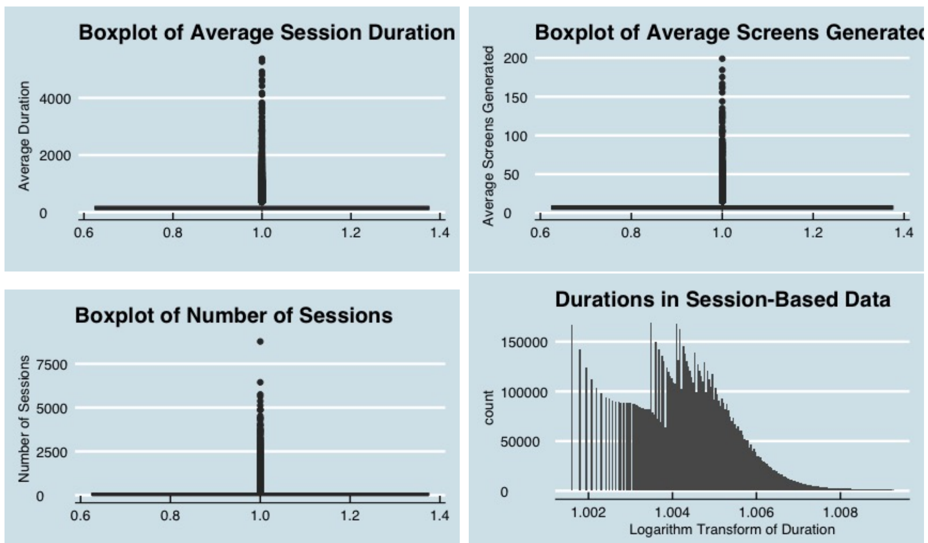

Figure 3

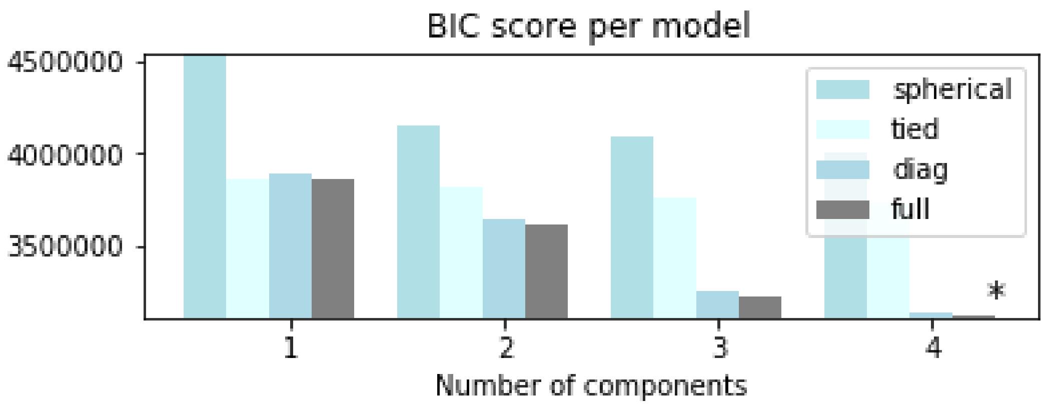

Figure 3.1:

After cleaning the data from several extreme observations, multivariate Gaussian Mixture Model was fitted to the data with Expectation Maximization algorithm used to estimate the mixture model's parameters. As the initial number of components was unknown, several different models were fitted in the range of 1 to 5 components and BIC (Bayesian Information Criterion) was used for optimal number of components' selection along with the best co-variance constrain (spherical, diagonal, tied or full covariance).

As seen in Figure 3.1, a final model with 4 components and full covariance was chosen which resulted in four user subgroups with the following characteristics:

Average Characteristics of User Subgroups						
User Group	Duration	Num. of Screens	Num. of Sessions	RFM Score	IAT	Users in the group
Group 1	172.52	7.57	367.30	333.0	40372	17923
Group 2	142.15	6.71	182.78	251.91	62352.8	8313
Group 3	131.68	6.40	23.87	275.63	649615	54000
Group 4	425.15	15.05	12.03	277.43	869347	9188

Interestingly, apart from the expected high, low and medium consuming users, a fourth user subgroup was distinguished by Gaussian Mixture Models thus resulting in:

- **Consistent Users:** users with many sessions of adequate duration and screen count as well as high RFM score and low session inter arrival times.
- **Moderate Users:** users having having moderate amount of sessions with corresponding duration and a moderate RFM score.

- **Average Users:** users having less average records but, surprisingly, high RFM score.

- **Heavy Users:** users having less but sessions that are considerably longer with more screens generated but with used within larger time intervals.

Although Group 2 and Group 3 do not differ in terms of many characteristics, however the radical difference between the number of sessions would result in Group 2 users being treated as outliers have they been part of Group 3 users which they are not, as they resulted in forming a completely independent users' subgroup. After the identification of user subgroups, outlier detection was done. As either the features or their logarithm transforms followed normal distribution (Figure 3.2), values being more than 3 standard deviations away from the mean were replaced with the mean value of the feature.

Once the data set was cleaned considering the subgroup characteristics, the four were then merged back and Decision Tree learning was used in order to understand what behavioral pattern leads the users to fall into this or that user subgroup. Random Forest classifier was used in order to understand which features play an important role in user classification. As presented in Figure 3.3, average duration, number of sessions, F_score, RFM and number of generated screens are among the important variables. Surprisingly, variable crash_rate does not have much importance. This might either be a result of very low number of crashes in the dataset or have an interpretation of crashes not having impact on overall user experience (which is highly debatable). Decision Trees rated RFM to be the most important feature followed by number of sessions, F_score, total duration and crash_rate being the last in the list. As for the classification results, both Decision Trees and Random Forest performed well with corresponding accuracy of 0.9822 and 0.9987. In a real world business scenario, the dataset will be constantly updated resulting in subgroups' characteristics' modification and distribution parameters' update. This, on its turn, will result in generating new rules and causing modifications in classification criteria and maybe even variable importance.

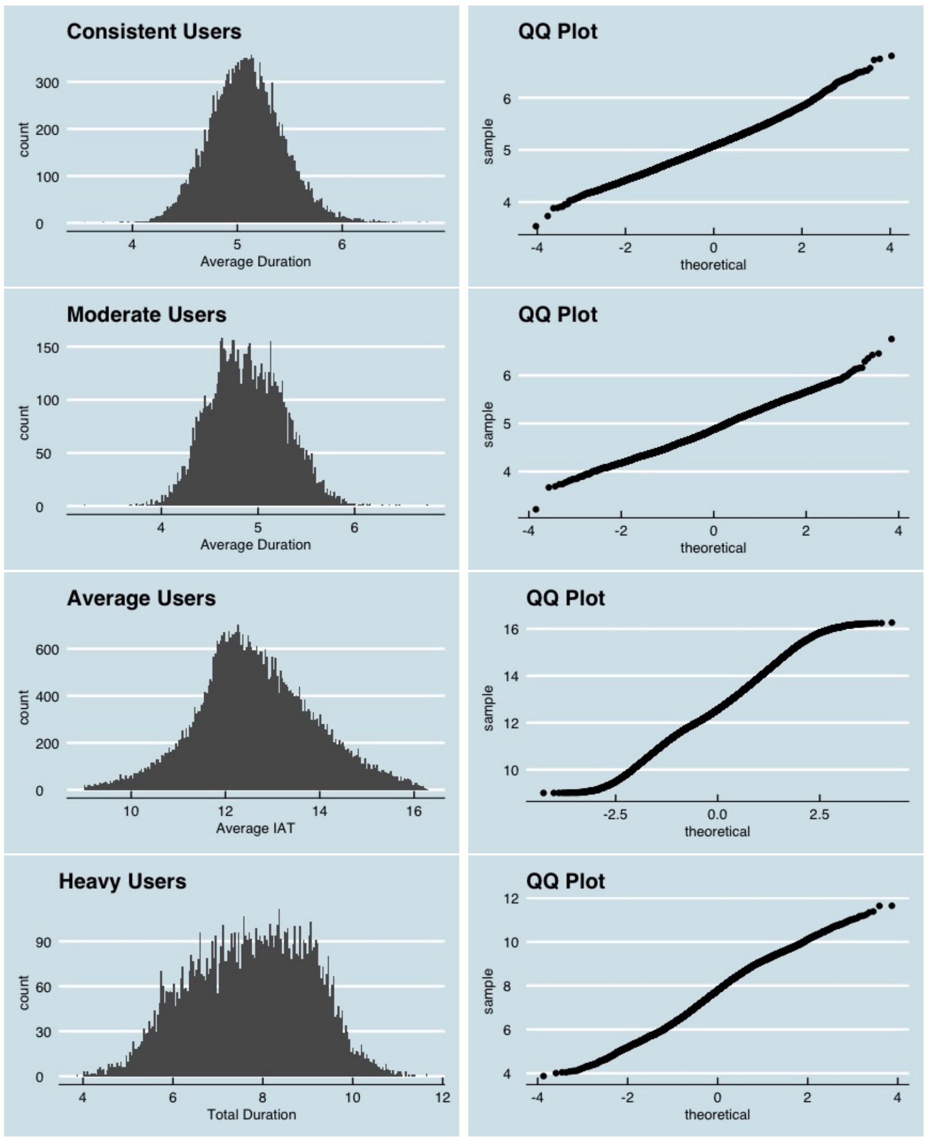

Figure 3.2: Normality Check Example From Each Subgroup

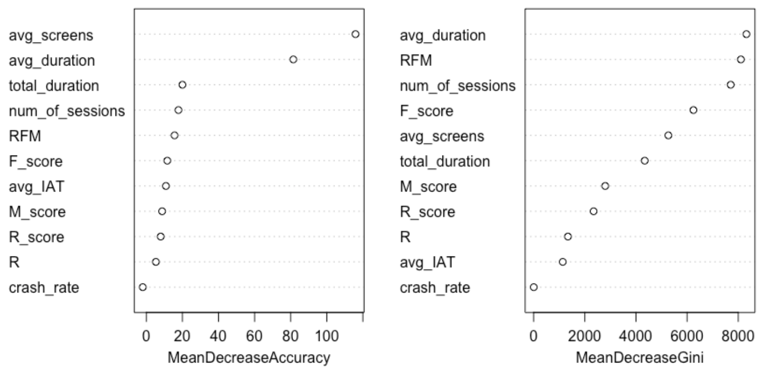

Figure 3.3: Variable importance in user classification (Random Forest)

3.2 Definition and Prediction of Customer Churn

As it has already been mentioned, application developers are unable to retrieve information about the app deletion, therefore it is impossible to state that this or that user has definitely deleted the app. To that end, having the definition of an Alternating Renewal Process in mind (Chapter 2), this section will focus on the description of application usage session as an alternating renewal process and suggest churn definition based on Renewal Process Theory.
An alternating renewal process alternates between two states *Up* and *Down* having a sequence of times system being up $\{U_n, n \geq 1\}$ and times system being down $\{D_n, n \geq 1\}$. In the context of this problem, user sessions can be described as an alternating renewal process where $\{U_n, n \geq 1\}$ represents the times a session is active, i.e. the session duration sequence and $\{D_n, n \geq 1\}$ represents the times a session is inactive, i.e. the inter arrival times. All assumptions regarding the process are being held, as if U is a generic up time and D is a generic down time, then the distribution of U does not have to be the the same as the distribution of D. Moreover, the pairs $\{(U_n, D_n), n \geq 1\}$ are i.i.d random vectors.

Define $Z(t)$ as the state at time t; i.e., $Z(t) = 1$ if the alternating renewal process is up at time t, starting at time 0 at the beginning of an up period; otherwise let $Z(t) = 0$. Applying renewal reward theorem deduces that the long-run proportion of time that the alternating renewal process is up is:

$$\text{long-run proportion up} = \lim_{x\to\infty} \frac{1}{t}\int_0^t Z(s)ds = \frac{E[U]}{E[U]+E[D]}$$

$$\lim_{x\to\infty} P(Z(t) = 1) = \frac{E[U]}{E[U]+E[D]}$$
$$\lim_{x\to\infty} P(Z(t) = 0) = \frac{E[D]}{E[U]+E[D]}$$

Thus, the probability of the session being inactive in the long run can be calculated by:

$$\frac{averageIAT}{averageIAT + averageduration}$$

and a user can be considered to have churned if:

$$\frac{Recency}{Recency + Lastsessionduration} \geq 1.1 \, \frac{averageIAT}{averageIAT + averageduration}$$

i.e. the proportion of time in which session is inactive exceeds the long-run proportion by 10%.

Labeling the data set by this definition resulted in marking 5 percent of the users as churned users which seems to be an adequate indication considering the overall app characteristics. Another approach to mark user churn that was tested on the dataset was calculating the maximum inter arrival time a user has ever had and defining churn if the recency exceeds that time by 10%. This, however, resulted in considering more than 50% of the users to have churned, which would be an extreme approach. Labeling results mirrored the overall behavior of user subgroups as only 27 users were marked as churn in **Consistent** users' subgroup, around 2600 users from the **Average** users and 866 from the **Heavy** users. During the user segmentation stage of the project it was clear that classification models cannot be generalized on the whole dataset and individual approach should be implemented on each user subgroup taking into consideration their different behavior and characteristics. Hence, classification models were built for each user subgroup individually. 30% of the users in each subgroup was kept as a testing set, while maintaining the proportion of target variable classes. The other 70% of the data was used for training and validation purposes.

Having a highly imbalanced data in the case of consistent and average users' subgroup, SMOTE oversampling technique was applied on the training data to generate synthetic observations and balance the proportion of feature classes. KNN, Naive Bayes and SVM classifiers were applied for this particular classification problem. In case of KNN algorithm, different features were included in the algorithm for different user subgroups. The number of optimal neighbors was also different for each subgroup (5 for consistent, 4 for moderate, 11 for average and 5 for heavy). As for the Naive Bayes classifier, only the variables for which the normality assumption was held were included in the model. Support vector machine performance was worse than expected in terms of sensitivity. The model did not handle data imbalance very well for any user subgroup, therefore SMOTE oversampling was applied to all of the subgroups. Even after oversampling, SVM gave satisfactory results in the case of heavy users only, all other subgroups had sensitivity less than 0.5. Figure 3.4 represents the results obtained by the classifiers for each user group in terms of accuracy, sensitivity and specificity. As one cannot make conclusions based only on the accuracy of the model, the more performance metrics to compare, the better. To that end, consider Youden's J statistic given as:

$$J = \text{sensitivity} + \text{specificity} - 1$$

Such measure considers both sensitivity and specificity of the model thus giving a more general evaluation. Computing J statistic for each model in each subgroup resulted in choosing the following best models with their corresponding Youden's index:

Consistent Users	KNN	0.964800
Moderate Users	Naive Bayes	0.998711
Average Users	KNN	0.993610
Heavy Users	KNN	0.988640

Such good performance of KNN can be justified intuitively. Even though the data consists of four different components, the user behaviour in each component is more or less the same. Therefore, classifying a user based other users that have the most similar behavior will have a better performance.

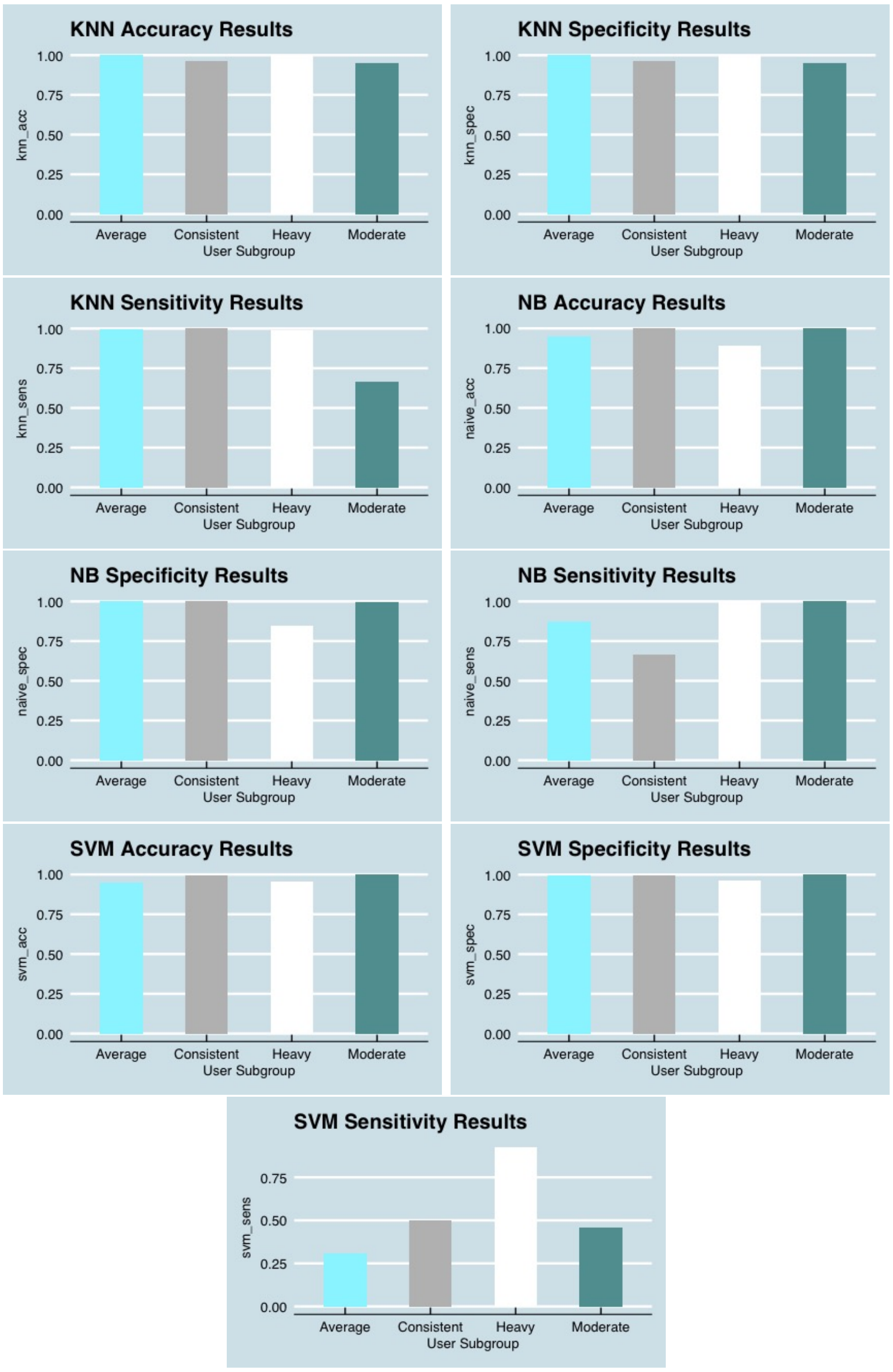

Figure 3.4: Classification Results

3.2.1 Conclusion

Customer segmentation has always been a useful tool in many business areas and now that mobile app development industry is at its peak, integration of customer segmentation in this field can considerably improve revenues. One of the approaches of mobile app user segmentation is analysis of user behavior data tracked while customers use the application. Having a dataset of 10.000.000 sessions, this project focused on derivation of user-based dataset that was later used in order to find user subgroups in the total user population. Statistical and Machine Learning tools such as Gaussian Mixture Models, Renewal Process Theory and classification algorithms were used in order to handle the objectives of the project.

As a result, four sub-populations were identified and models were developed to identify customer at risk of churn. In a real life business scenario, the application of this project will be the following: having a constantly flowing data, the parameters of developed models will be constantly updated and adjusted to the current trends. Users will be constantly categorized between the subgroups and tested whether they belong to the risk group or not and prevention strategies will be applied accordingly in order to save the users.

Further research of the project can include the expansion of the data. Features including information about the touches, buttons, and app design can have great contributive value to this project and help to understand the behavior leading to churn way better.

Bibliography

[1] Worldwide Mobile App Revenues. *The Statistical Portal*.
Retrieved from:
`https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/`

[2] Divya D. Nimbalkar, Asst Prof. Paulami Shah, *Data mining using RFM Analysis*. International Journal of Scientific and Engineering Research, Volume 4, Issue 12, December, 2013, ISSN 2229-5518.
Retrieved from:
`https://pdfs.semanticscholar.org/5aa6/bcb19728998ff6f97cb68ce9e9670293be97.pdf`

[3] Khajvand M., Zolfaghar K., Ashoori S., Alizadeh S., *Estimating customer lifetime value based on RFM analysis of customer purchase behavior: Case study*, Procedia Computer Science, Volume 3, 2011, pp. 57-63.
Retrieved from:
`https://doi.org/10.1016/j.procs.2010.12.011`

[4] Bishop C.M., *Pattern Recognition and Machine Learning*, 2006, pp. 430-435, ISBN-13: 978-0387-31073-2.

[5] Author NA, *The Bayes Information Criterion*, Massachusetts Institute of Technology, Dec. 2015.
Retrieved from:
`http://www-math.mit.edu/~rmd/650/bic.pdf`

[6] Erar B., *Mixture model cluster analysis under different covariance structures using information complexity*, University of Tennessee, Knoxville, 2011.
Retieved from:
`http://trace.tennessee.edu/cgi/viewcontent.cgi?article=2096&context=utk_gradthes`

[7] Russel S., Norvig P., *Artificial Intelligence, a Modern Approach, third edition.*, ISBN-13: 978-0-13-604259-4

[8] Fristedt B., Gray L., *A Modern Approach to Probability Theory.*, Birkhauser, Boston, MA, 1997.

[9] Sigman K., *Introduction to Renewal Theory*, Columbia University, 2009.
Retrieved from:
`http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-RRT.pdf`

[10] Whitt A., *Alternating Renewal Processes and The Renewal Equation*, Columbia University, 2013.
Retrieved from:
`http://www.columbia.edu/~ww2040/3106F13/lect1119.pdf`

[11] Chawla N.,Bowyer K.,HallL.,Kegelmeyer P. *SMOTE:SyntheticMinor- ity Over-sampling Technique*,Journal of Artificial Intelligence Research 16 (2002) 321–357.
Retrieved from:
`https://arxiv.org/pdf/1106.1813.pdf`

[12] Dullaghan C., Rozaki E., *Integration of Machine Learning Techniques To Evaluate Dynamic Customer Segmentation Analysis for Mobile Customers*, International Journal of Data Mining Knowledge Management Process (IJDKP) Vol.7, No.1, January 2017.
Retrieved from:
`https://arxiv.org/pdf/1702.02215.pdf`