How to predict the one-year survival of mobile apps?

-- Evidence from gaming apps on Google Play Store

Introduction

Mobile app economy worldwide grows from 1.3 trillion US dollars in 2016 to 3.35 trillion US dollars in 2019 and is expected to further expand to reach 6.3 trillion US dollars in 2021 (Statista, 2020). Solely in the US, mobile app economy represents 558.34 billion dollars in 2018 (Deloitte, 2018). As the entry barrier and transaction cost decrease, the competition in mobile app market increases. There are over 300 thousand companies active in this market in the US only in 2018 and there are 2.8 and 2.2 million apps available for download on Apple Store and Google Play Store in 2019 (Deloitte, 2018; Blair, 2019). Intensified competition in turn makes it hard for apps to survive in market: nearly 90% of apps are zombie apps that people barely download and use and only 0.1% of apps can achieve financial success by 2018 (Adjust, 2017; Daisyme, 2018). Given the large chance of failure, it is critical for investors and developers who are interested in mobile app market to understand which factors determine the success or failure of an app before they extensively invest money and energy in app projects. With this knowledge, they can adjust their app design and marketing mix for the app to achieve the best outcome. This piece intends to use a dataset of over 2000 gaming apps on Googly Play Store to answer this question as well as to create a decision tree to help investors and developers make quick decisions in practice. However, given the limitation of the dataset, there are a few caveats regarding the estimations that will be discussed shortly.

Data

The dataset records 2579 game apps introduced to Google Play Store in 2014 over a one-year period (i.e., 356 days). It records whether each app was officially removed from the store during this one-year period in a binary form (i.e., 0 for no, 1 for yes) and nearly half of them (i.e., 1119 apps) were officially removed from the app. It also records the app features of each app (e.g., display language, maturity level, installation size), marketing mix of each app (e.g., pricing, adverting), developer characteristics of each app and user feedback of each app (e.g., average and standard deviation of app rating). Table 1 describes the 10 predictors and 2 control variables in this dataset. Since the dataset does not include information about financial performance of each app, success (vs failure) of an app is defined as being present (vs removed from) the app store **over one-year period**. Therefore, the question seems to be boiled down to a survival question and thus survival model (e.g., AFT, Cox or discrete-time survival model) should be most appropriate. Unfortunately, the dataset misses the information about survival time of each app which should be the key dependent variable for AFT model and Cox regression nor the information about baseline hazard which is necessary for discrete-time survival model. Instead, I will use logistic model. Censorship is usually a big concern in this case, however, to address the right censorship problem, I strictly state that my focal dependent variable is one-year survival/death which is how success/failure is defined and I will not generalize it to general survival/death. As for the left censorship problem, neither survival nor logistical model can address it, but it may not be a big issue given the sampling is random and there is no strong reason to assume apps in this sample are less or more likely to survive or die within one year than apps died before 2014. In addition, since all apps are gaming apps, I would be cautious to generalize any findings to other app categories unless there is no category-level effect.

Table 1

App Features	
APP_ISPAID	1 if the app is a paid app and 0 if it is a free app
INAPP_PURCHASE	1 if the app has in-app purchase and 0 otherwise
CONTENT_TYPE	1, 2, 3, 4 if the content rating is everyone, low maturity, medium maturity and high maturity; they are treated as continuous rather than categorical and higher number indicates higher maturity

APP_ENGLISH	1 if the app displays English and 0 if it displays in other languages; control variable
INSTALL_SIZE	the size required to install the app in MB on last day of observation; control variable
Marketing Mix	
APP_DESCRIPTION	the number of words in the app description
APP_IMAGE	the number of images available in the app description
CUM_TOP	the cumulative number of times that the app landing on various top charts of Google Play
Developer information	
TOP_DEVELOPER	if the developer of the app is selected as top developer and 0 otherwise
APP_DEVNUM	the total number of apps that the developer of the app owns on Google Play on the last day of observation
User feedback	
RATING_AVG	the mean of app rating
RATING_DEV	the standard deviation of the app rating
APP_DEAD	1 if the app is dead and 0 otherwise

Theoretical Background

There are two papers that are most relevant to this piece. First, Jung, Beak and Lee (2012) used AFT model to study the factors that affected the survival of free apps and paid apps with a dataset of 100 top free and paid apps in a Korea App store; they found that user ratings, contents size and early entrant advantage had stronger positive effects on the survival of free apps than of paid apps. Lee and Raghu (2012) also studied the same question but with a proportional hazard model and a different dataset of 300 top apps on App Store; they found that diversification, user rating, initial ranking and free offerings had positive effects on the survival of apps. Interestingly, both of them selected a sample based on the position of the apps on the app store (i.e., top 100, top 300); however, neither addressed the selection bias caused by their selection procedure. Different from their dataset, apps in this dataset are randomly selected regardless of their position but only from one category (i.e., gaming). In addition, some of the predictors in this dataset are different from their datasets. Table 2 lists the hypotheses and the corresponding rationales.

Table 2

App Features	
H1a: free apps are more likely to survive over a one-year period than paid apps, because they can have larger user base.	H1b: free apps are less likely to survive over a one-year period than paid apps, because they have worse cash flow.
H2a: apps with in-app purchase are more likely to survive over a one-year period than those without in-app purchase, because they can have better cash flow.	H2b: apps with in-app purchase are less likely to survive over a one-year period than those without in-app purchase, because users may abandon these apps when they are asked to make in-app purchase.
H3: apps with less mature contents are more likely to survive within one year, because they can have larger user base.	
*Default language and install size of the apps are not expected to have any influence.	
Marketing Mix	
H4a&b: apps with more comprehensive app description (more words and more images in app description) are more likely to survive over a one-year period, because users are more likely to download these apps and their expectations of the apps are more accurate that leads to higher satisfaction.	
H5: apps with higher times of landing in the top chart are more likely to survive over a one-year period, because they are more visible to users.	
Developer information	
H6: apps developed by top developers are more likely to survive over a one-year period than apps not developed by top developers, because their quality tend to be higher.	
H7: apps developed by developers who have more apps in app store are less likely to survive over aone-year period, because these developers may pay less attention to maintain and update their apps and in turn retain fewer users.	
User feedback	

H8: apps with higher average rating are more likely to survive over a one-year period, because their quality tends to be higher.

(I also create an interaction term to directly probe the interaction between the average rating and the type of apps (i.e., free vs paid) suggested by Jung, Beak and Lee (2012)) who conducted AFT models for two sub-datasets (i.e., top 100 free apps and top 100 paid apps) separately.)

H9: apps with higher standard deviation of ratings are less to survive over a one-year period, because they are niche apps that have smaller consumer base.

H10: there is an interaction between the average rating and the type of in-app purchase (i.e., with vs without): the positive effect of high rating is stronger among apps without in-app purchase while the negative effect of low rating is stronger among apps with in-app purchase.

H11: there is an interaction between the standard deviation of ratings and the type of apps: free apps are likely to survive over a one-year period if it is a niche app than if it is not a niche app, because they have smaller user base.

Models and Results

Model-free Analysis

Out of 2579 apps in this sample, 1866 apps are free apps and 713 are paid apps; 1968 apps do not have in-app purchase and 611 apps have in-app purchase; 2539 apps are displayed in English and 40 apps are displayed in other languages; 99 apps are developed by top developers and 2480 apps are not developed by top developers; 1387 apps have "everyone" level contents, 653 apps have "low maturity" level contents, 471 apps have "medium maturity" level contents and 68 apps have "high maturity" level contents (when it is treated as continuous variable, the mean is 1.70). In addition, 1119 apps and 1460 apps survived died during the one-year period. In this sample, the average install size is 28.15 MB (SD = 97.19) and the average number of words and images in app description is 6.59 (SD = 6.59) and 13.81 (SD = 6.71). The developers of the apps have 14.64 apps on average (SD = 37.5), suggesting that most developers and experienced developers. The average rating of the apps is 2.80 out of 5.00 (SD = 0.21)), suggesting that most apps are relatively low-quality apps. The standard deviation of the rating of the apps is 1.69 (SD = 0.35), suggesting that most apps are mainstream apps. The average times of landing in top chart of the apps are 8.91 (SD = 0.41, Range = [5.00, 9.00], suggesting that the visibility of the apps is similar.

Chi-squared tests suggest that a larger proportion of free apps died within one year (46% of free apps died and 35% of paid apps died; Chi-squared = 28, df =1, p < 0.001) and a larger proportion of apps without in-app purchases died within one year (49% of free apps died and 21% of paid apps died; Chi-squared = 111, df =1, p < 0.001), supporting H1b and H2a and contradicting to H1a and H2b. Chi-squared test also suggests that a smaller proportion of apps developed by top developers died within one year (21% apps developed by top developers died and 44% of apps not developed by top developers died; Chi-squared = 20, df =1, p < 0.001), supporting H6. Moreover, consistent with expectation, almost equal proportion of apps display in English (43%) and in other languages died within one year (40%) (Chi-squared = 0.08, df =1, p < 0.78). Since correlation tests cannot probe the non-linear relationships between continuous predictors and APP_DEAD, I treat APP_DEAD as a group factor and explore whether there is difference in the means of continuous predictors between groups (i.e., 0 vs 1). Table 3 shows the results. These results suggest that on average dead apps had fewer words and images in description and higher standard deviation of ratings, supporting H4ab and H9). Contrary to H7, developers of dead apps have fewer apps in Google Play. Surprisingly, dead apps and survived apps had almost identical average ratings and times of landing in top charts, contradicting to H5 and H8. It is also unexpected that dead apps had smaller install size. These model-free analysis results provide preliminary evidence for some hypotheses; however, rigorous model-based analysis is needed.

Table 3

	APP_DEAD	MEAN (SD)	Levene's test	T-test (*equal variance assumed)
INSTALL	0	34 (77)	.018	.002
_SIZE	1	21 (117)		
APP_	0	7.3 (9.2)	<.001	<.001

DES	1	5.6 (7.4)		
APP_ IMAGE	0	15.5 (6.4)	.338	<.001*
	1	11.7 (6.5)		
APP_ DEVNUM	0	18.6 (40.6)	<.001	<.001
	1	9.51 (32.3)		
RATING_ AVE	0	2.80 (0.19)	.001	0.753
	1	2.80 (0.23)		
RATING_ DEV	0	1.67 (0.31)	<.001	<.001
	1	1.74 (0.38)		
CUM_TOP	0	8.92 (0.37)	.003	0.151
	1	8.89 (0.46)		

Logistic Regression

The baseline logistic model is $\log\left(\frac{p}{1-p}\right) = \alpha + \beta\boldsymbol{x}$ where $p = P(APP\_DEAD = 1)$ and $\boldsymbol{x}$ is a vector of variables described in Table 1 and three interaction terms (i.e., RATING_AVG * APP_ISPAID, RATING_AVG * INAPP_PURCHASE and RATING_DEV * APP_ISPAID). Although the display language and install size are not expected to affect APP_DEAD, they are included in the baseline model as the control variables. Since all observations are independent from each other (i.e., no repeated measures or matching), the dependent variable is binary by nature, the sample size is decent, and the linear relationships between variables and log odds are assumed, the biggest concern regarding the assumptions of logistics regression here is multicollinearity. Fortunately, it is not a severe issue in the baseline model. Although VIF of APP_ISPAID, INAPP_PURCHASE, AveRatingPaid, AveRatingPurchase (VIFs > 150) and DevRatingPaid (VIF = 38) are high, these high VIFs are caused by the interaction terms and thus can be ignored (Allison, 2012). When the interaction terms are deleted, VIFs of all variables are below 1.5. I then conduct stepwise model selection and the model selected by stepwise model selection does not significantly improve AIC ($AIC_{baseline}$ =3147.3 $AIC_{selected}$ =3141). The selected model is also biased and should be ignored, because the high VIF variables should not be deleted. All three interaction terms in the baseline model are not significant ($\boldsymbol{p}$s > 0.14) but produce some noises (e.g., high VIF); therefore, I delete them in the final model

(AIC_{final} = 3145, VIFs < 1.5) and I make inferences from the final model. Table 4 shows the model estimates and odds ratio of each predictor.

The results suggest that paid apps and apps with in-app-purchase are 59% less likely to die over a one-year period than free apps and apps without in-app purchase, supporting H1b and H2a and consistent with the model-free analysis results. However, the content type does not affect the survival /death of the apps in a one-year period not supporting H3. Apps with one more word and one more image in app description are 2% and 7% less likely to die over a one-year period, supporting H4ab and consistent with the model-free analysis results. Apps with one more time of landing the top chart is 40% less likely to die over a one-year period, supporting H5. Apps with one-star higher rating (i.e., apps with higher quality) and one-unit higher standard deviation (i.e., niche apps) are 68% less likely and 130% more likely to die over a one-year period, supporting H8 and H9. As for the impact of developer characteristics, whether the apps are developed by top developers or not does not affect the survival/death of the apps over a one-year period, not supporting H6; however, apps developed by developers who have one more app in Google Play are 1% less likely to die over a one-year period, supporting H7. Surprisingly, apps that displays in English is 66% less likely to die over a one-year period, probably due to the larger user base since most users of Google Play may not understand other languages. Table 5 summarizes whether each hypothesis is supported or not.

Table 4

	Estimates (p-value)	Odds Ratio (for significant predictors only)
Intercept	6.91 (<.001)	NA
APP_ISPAID	-0.89 (<.001)	0.41
INAPP_PURCHASE	-0.89 (<.001)	0.41
TOP_DEVELOPER	-0.23 (.425)	NA
CONTENT_TYPE	-0.05 (.313)	NA
APP_ENGLISH	-1.07 (.010)	0.34
INSTALL_SIZE	0.0005 (.326)	NA
APP_DESCRIPTION	-0.02 (.002)	0.98
APP_IMAGE	-0.07 (<.001)	0.93
APP_DEVNUM	-0.01 (<.001)	0.99

RATING_AVG	-0.47 (0.044)	0.32
RATING_DEV	0.83 (<.001)	2.30
CUM_TOP	-0.50 (<.001)	0.60

Table 5

H1a	H1b	H2a	H2b	H3	H4ab
No	Yes	Yes	No	No	Yes
H5	H6	H7	H8	H9	H10/11
Yes	No	Yes	Yes	Yes	No

Decision Tree

I first conduct the decision tree analysis with all variables described in Table 1. Based on the 1SE rule, I select a tree with 10 splits with a cross-validation error at 0.69, and it yields an accuracy at 0.72 (Tree 1; Figure 1). I then conduct the same analysis only with significant variables in logistic regression analysis. Based on the 1SE rule, I select a tree with 11 splits with a cross-validation error at 0.68 and it yields an accuracy at 0.73 (Tree 2; Figure 2). Table 7a and 7b are the confusion matrixes Tree 1 and Tree 2. In practice, investors and developers can use either tree depending on the information they have at hand, because the trees are equally good from a statistical perspective. However, Tree 1 (vs Tree 2) is preferred if they are more interested in predicting the death (vs survival) of apps, because Tree 1(vs Tree 2) is 4% (vs 3.5%) more accurate in predicting the death (vs survival) of apps.

I further compare the prediction ability of decision tree to that of other classification tools (i.e., logistic regression, discriminant analysis and support vector machine) with the same dataset. Table 6 shows the accuracy of each classification tool and Figure 3 depicts the ROC of the final logistic regression model (AUC = 0.73). These results suggest that decision tree outperforms logistic regression and discriminant analysis but performs as well as SVM. Therefore, it should be adopted. However, the overall prediction ability is only moderate and other factors should be found to improve the classification.

Table 6

Decision Tree	Logistic Regression	Discriminant Analysis	SVM
0.73	0.67	0.67	0.92

Figure 1

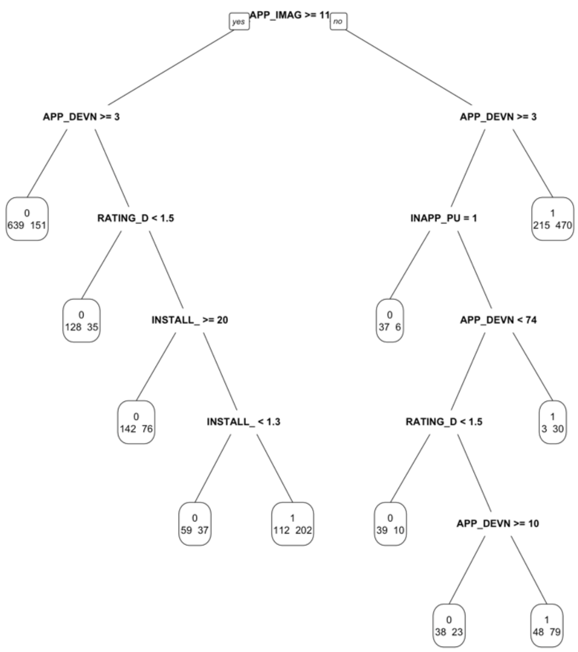

Figure 2

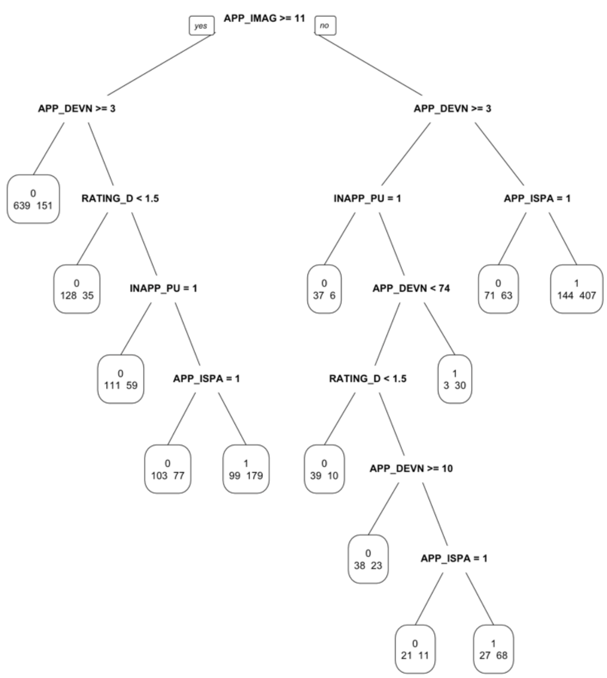

Figure 3

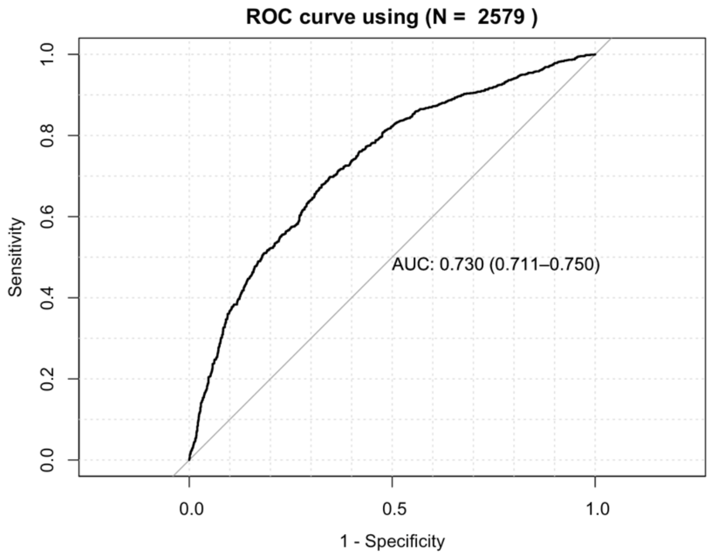

Table 7a

	Predicted Survival	Predicted Death
Actual Survival	1082	338
Actual Death	378	781

Table 7b

	Predicted Survival	Predicted Death
Actual Survival	1187	435
Actual Death	273	684

Conclusion

This piece confirms several commonly held intuitions regarding the determinants of the success or failure of mobile apps. Using a dataset of 2579 mobile apps on Google Play store, I find that app features, marketing mix, developer characteristics and use feedback all influence the success of mobile apps. More specifically, in terms of app features, free apps, apps with in-store purchase, apps in English are more likely to survive over a one-year period; in terms of marketing mix, apps with more pictures and words in app descriptions and higher visuality are more likely to survive over a one-year period; in terms of developer characteristics, apps developed by more experienced developers are more likely to survive over a one-year period; in terms of user feedback, apps with higher average rating and lower standard deviation of rating are more likely to survive over a one-year period. In addition, I develop a decision tree with moderate prediction ability that can be used by investors and developers in practice to predict the potential of an app.

There are a few caveats. First, as discussed in the previous sections, logistic regression is not the best way to predict the survival or death of mobile apps; if the dataset is permitted, it is better to use survival models and uncover the baseline hazard function as well as to control for app-level or developer-level unobserved heterogeneity for a more unbiased and consistent estimation. Second, this dataset only speaks the situation of game apps, and it is necessary to redo the analysis with a dataset from other app categories. Third, the prediction ability of the decision tree is only moderate, and it is necessary to improve the model with other or more variables.

Reference

Statista (2020), Size of the App Economy Worldwide from 2016 to 2021. Retrieved from https://www.statista.com/statistics/267209/global-app-economy/

Deloitte (2018), The App Economy in the United States: A review of the mobile app market and its contribution to the United States Economy. Retrieved from https://www.ftc.gov/system/files/documents/public_comments/2018/08/ftc-2018-0048-d-0121-155299.pdf

Blair, I. (2019), Mobile App Download and Usage Statistics. Retrieved from https://buildfire.com/app-statistics/

Adjust (2017), The Zombie Uprising: A look at the undead App Store in 2016. Retrieved from http://learn.adjust.com/rs/108-GAZ-487/images/The_Zombie_uprising_2016.pdf?mkt_tok=eyJpIjoiTVdWbVpEaGpaRGxtTXpVMiIsInQiOiJ5Y1piQUpzUjh3QUlRdVNRcGRJSnJcLzQwWlFIRU5qSzkwUnFIb3A1cjhpMHdoWlpDSXhQeHJhZzBkOGZzY2hqd1FlSEdnbHV0ZTV5SVhrbnJnRmh3RDFSUlZ5SWJ6cURGbUNGQ0xCVmJtWGs9In0%3D

Daisyme, P. (2018), 9999 in 10000 Mobile Apps Fail: Here is why. Retrieved from https://www.startupgrind.com/blog/9999-in-10000-mobile-apps-will-fail-heres-why/

Allison, P. (2012), When Can You Safely Ignore Multicollinearity? Retrieved from https://statisticalhorizons.com/multicollinearity

Appendix

```
> #Data Process
> mydata <- cbind(ProjectData[3:18])
> View(mydata)
> dead<-mydata$APP_DEAD
> table(dead)
dead
   0    1
1460 1119
> describe(mydata)
                  vars    n  mean    sd median trimmed   mad  min     max   range  skew kurtosis   se
APP_ISPAID           1 2579  0.28  0.45   0.00    0.22  0.00 0.00    1.00    1.00  1.00    -1.00 0.01
INAPP_PURCHASE       2 2579  0.24  0.43   0.00    0.17  0.00 0.00    1.00    1.00  1.24    -0.47 0.01
TOP_DEVELOPER        3 2579  0.04  0.19   0.00    0.00  0.00 0.00    1.00    1.00  4.80    21.07 0.00
CONTENT_TYPE         4 2579  1.70  0.86   1.00    1.59  0.00 1.00    4.00    3.00  0.87    -0.41 0.02
APP_ENGLISH          5 2579  0.98  0.12   1.00    1.00  0.00 0.00    1.00    1.00 -7.84    59.44 0.00
INSTALL_SIZE         6 2579 28.15 97.19  14.02   16.94 16.31 0.00 3719.84 3719.84 23.75   823.18 1.91
APP_DESCRIPTION      7 2579  6.59  8.49   4.58    5.24  3.83 0.03  125.52  125.49  6.17    59.05 0.17
APP_IMAGE            8 2579 13.81  6.71  13.00   13.38  8.90 3.00   31.00   28.00  0.41    -0.87 0.13
APP_DEVNUM           9 2579 14.64 37.50   2.00    4.82  1.48 1.00  285.00  284.00  4.03    16.76 0.74
LAG_RATING_AVG      10 2579  2.80  0.21   2.79    2.79  0.11 2.00    4.85    2.85  2.49    16.28 0.00
LAG_RATING_DEV      11 2579  1.69  0.35   1.61    1.63  0.10 0.00    5.90    5.90  4.19    32.58 0.01
LAG_CUM_TOP         12 2579  8.91  0.41   9.00    9.00  0.00 5.00    9.00    4.00 -5.01    27.71 0.01
AveRatingPaid       13 2579  0.76  1.24   0.00    0.57  0.00 0.00    4.50    4.50  1.06    -0.77 0.02
AveRatingPurchase   14 2579  0.66  1.19   0.00    0.47  0.00 0.00    4.50    4.50  1.26    -0.39 0.02
DevRatingPaid       15 2579  0.38  0.70   0.00    0.26  0.00 0.00    5.20    5.20  1.44     1.00 0.01
APP_DEAD            16 2579  0.43  0.50   0.00    0.42  0.00 0.00    1.00    1.00  0.27    -1.93 0.01
> table(mydata$APP_ISPAID)

   0    1
1866  713
> table(mydata$INAPP_PURCHASE)

   0    1
1968  611
> table(mydata$APP_ENGLISH)

   0    1
  40 2539
> table(mydata$CONTENT_TYPE)

   1    2    3    4
1387  653  471   68
> table(mydata$TOP_DEVELOPER)

   0    1
2480   99
> #Model-free analysis
> #Chi-square tests
> table(mydata$APP_ISPAID,mydata$APP_DEAD)

      0   1
  0 996 870
  1 464 249
> prop.table(table(mydata$APP_ISPAID,mydata$APP_DEAD))

             0          1
  0 0.38619620 0.33734005
  1 0.17991470 0.09654905
> prop.table(table(mydata$APP_ISPAID,mydata$APP_DEAD),1)

            0         1
  0 0.5337621 0.4662379
  1 0.6507714 0.3492286
> chisq.test(mydata$APP_ISPAID,mydata$APP_DEAD)

	Pearson's Chi-squared test with Yates' continuity correction

data:  mydata$APP_ISPAID and mydata$APP_DEAD
X-squared = 28.28, df = 1, p-value = 1.05e-07

> table(mydata$INAPP_PURCHASE,mydata$APP_DEAD)

       0    1
  0 1001  967
  1  459  152
> prop.table(table(mydata$INAPP_PURCHASE,mydata$APP_DEAD))

             0          1
  0 0.38813494 0.37495153
  1 0.17797596 0.05893757
> prop.table(table(mydata$INAPP_PURCHASE,mydata$APP_DEAD),1)

            0         1
  0 0.5086382 0.4913618
  1 0.7512275 0.2487725
> chisq.test(mydata$INAPP_PURCHASE,mydata$APP_DEAD)

	Pearson's Chi-squared test with Yates' continuity correction

data:  mydata$INAPP_PURCHASE and mydata$APP_DEAD
X-squared = 110.72, df = 1, p-value < 2.2e-16
```

```
> table(mydata$TOP_DEVELOPER,mydata$APP_DEAD)

       0    1
  0 1382 1098
  1   78   21
> prop.table(table(mydata$TOP_DEVELOPER,mydata$APP_DEAD))

              0           1
  0 0.535866615 0.425746413
  1 0.030244281 0.008142691
> prop.table(table(mydata$TOP_DEVELOPER,mydata$APP_DEAD),1)

            0         1
  0 0.5572581 0.4427419
  1 0.7878788 0.2121212
> chisq.test(mydata$TOP_DEVELOPER,mydata$APP_DEAD)

	Pearson's Chi-squared test with Yates' continuity correction

data:  mydata$TOP_DEVELOPER and mydata$APP_DEAD
X-squared = 19.685, df = 1, p-value = 9.13e-06
> table(mydata$CONTENT_TYPE,mydata$APP_DEAD)

      0   1
  1 776 611
  2 352 301
  3 300 171
  4  32  36
> prop.table(table(mydata$CONTENT_TYPE,mydata$APP_DEAD))

              0          1
  1 0.30089182 0.23691353
  2 0.13648701 0.11671190
  3 0.11632416 0.06630477
  4 0.01240791 0.01395890
> prop.table(table(mydata$CONTENT_TYPE,mydata$APP_DEAD),1)

            0         1
  1 0.5594809 0.4405191
  2 0.5390505 0.4609495
  3 0.6369427 0.3630573
  4 0.4705882 0.5294118
> chisq.test(mydata$CONTENT_TYPE,mydata$APP_DEAD)

	Pearson's Chi-squared test

data:  mydata$CONTENT_TYPE and mydata$APP_DEAD
X-squared = 14.341, df = 3, p-value = 0.002475
```

```
> table(mydata$APP_ENGLISH,mydata$APP_DEAD)

       0    1
  0   24   16
  1 1436 1103
> prop.table(table(mydata$APP_ENGLISH,mydata$APP_DEAD))

              0           1
  0 0.009305933 0.006203955
  1 0.556804963 0.427685149
> prop.table(table(mydata$APP_ENGLISH,mydata$APP_DEAD),1)

           0        1
  0 0.600000 0.400000
  1 0.565577 0.434423
> chisq.test(mydata$APP_ENGLISH,mydata$APP_DEAD)

	Pearson's Chi-squared test with Yates' continuity correction

data:  mydata$APP_ENGLISH and mydata$APP_DEAD
X-squared = 0.075675, df = 1, p-value = 0.7832
```

Group Statistics

	APP_DEAD	N	Mean	Std. Deviation	Std. Error Mean
INSTALL_SIZE	0	1460	33.6233748	77.3739900	2.02496987
	1	1119	21.0146163	117.801489	3.52156524
APP_DESCRIPTION	0	1460	7.3496	9.19760	.24071
	1	1119	5.6001	7.36392	.22014
APP_IMAGE	0	1460	15.47	6.383	.167
	1	1119	11.66	6.505	.194
APP_DEVNUM	0	1460	18.58	40.603	1.063
	1	1119	9.51	32.328	.966
RATING_AVG	0	1460	2.79846068	.191239513	.005004967
	1	1119	2.80112250	.228359012	.006826579
RATING_DEV	0	1460	1.66080859	.314501425	.008230878
	1	1119	1.73664741	.384331622	.011489234
CUM_TOP	0	1460	8.92	.370	.010
	1	1119	8.89	.460	.014

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
INSTALL_SIZE	Equal variances assumed	5.629	.018	3.271	2577	.001	12.6087585	3.85413564	5.05124185	20.1662751
	Equal variances not assumed			3.104	1826.538	.002	12.6087585	4.06225611	4.64160340	20.5759136
APP_DESCRIPTION	Equal variances assumed	12.586	.000	5.210	2577	.000	1.74949	.33577	1.09108	2.40791
	Equal variances not assumed			5.363	2572.112	.000	1.74949	.32619	1.10986	2.38912
APP_IMAGE	Equal variances assumed	.917	.338	14.887	2577	.000	3.807	.256	3.305	4.308
	Equal variances not assumed			14.850	2382.534	.000	3.807	.256	3.304	4.310
APP_DEVNUM	Equal variances assumed	62.688	.000	6.126	2577	.000	9.065	1.480	6.163	11.966
	Equal variances not assumed			6.311	2573.277	.000	9.065	1.436	6.248	11.881
RATING_AVG	Equal variances assumed	10.191	.001	-.322	2577	.748	-.00266182	.008270410	-.01887914	.013555504
	Equal variances not assumed			-.314	2163.846	.753	-.00266182	.008464743	-.01926170	.013938058
RATING_DEV	Equal variances assumed	16.181	.000	-5.508	2577	.000	-.07583881	.013768089	-.10283645	-.04884118
	Equal variances not assumed			-5.366	2130.119	.000	-.07583881	.014133289	-.10355530	-.04812233
CUM_TOP	Equal variances assumed	9.056	.003	1.477	2577	.140	.024	.016	-.008	.056
	Equal variances not assumed			1.436	2107.181	.151	.024	.017	-.009	.057

```
> #Logistic Regression
> logistic <- glm(APP_DEAD ~. , family=binomial(link="logit"), na.action=na.pass, data=mydata, )
> summary(logistic)

Call:
glm(formula = APP_DEAD ~ ., family = binomial(link = "logit"),
    data = mydata, na.action = na.pass)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1545  -0.9616  -0.5873   1.0144   2.6574

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        8.0182117  1.5732208   5.097 3.46e-07 ***
APP_ISPAID        -2.9143018  1.3692532  -2.128  0.03330 *
INAPP_PURCHASE    -0.4874535  1.9553368  -0.249  0.80313
TOP_DEVELOPER     -0.2169151  0.2848371  -0.762  0.44633
CONTENT_TYPE      -0.0521064  0.0515491  -1.011  0.31211
APP_ENGLISH       -1.0410986  0.4167998  -2.498  0.01250 *
INSTALL_SIZE       0.0004474  0.0004641   0.964  0.33508
APP_DESCRIPTION   -0.0213679  0.0067781  -3.152  0.00162 **
APP_IMAGE         -0.0743387  0.0070261 -10.580  < 2e-16 ***
APP_DEVNUM        -0.0067258  0.0012993  -5.176 2.26e-07 ***
RATING_AVG        -0.8742345  0.3948511  -2.214  0.02682 *
RATING_DEV         0.8292295  0.1624146   5.106 3.30e-07 ***
CUM_TOP           -0.4998998  0.1147504  -4.356 1.32e-05 ***
AveRatingPaid      0.7241753  0.4864251   1.489  0.13655
AveRatingPurchase -0.4329816  0.6995111  -0.619  0.53593
DevRatingPaid      0.4850109  0.4050621   1.197  0.23116
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3530.0  on 2578  degrees of freedom
Residual deviance: 3115.3  on 2563  degrees of freedom
AIC: 3147.3

Number of Fisher Scoring iterations: 4

> #VIF
> vif(logistic)
       APP_ISPAID    INAPP_PURCHASE     TOP_DEVELOPER      CONTENT_TYPE       APP_ENGLISH
       196.306430        322.707665          1.162913          1.055852          1.381225
     INSTALL_SIZE   APP_DESCRIPTION         APP_IMAGE        APP_DEVNUM        RATING_AVG
         1.091992          1.376275          1.087645          1.023607          3.257388
       RATING_DEV           CUM_TOP     AveRatingPaid AveRatingPurchase     DevRatingPaid
         1.480483          1.099550        185.616583        321.691363         37.973699
> vif(logistic2)
     APP_ISPAID  INAPP_PURCHASE   TOP_DEVELOPER    CONTENT_TYPE     APP_ENGLISH    INSTALL_SIZE
       1.381772        1.132099        1.163198        1.053904        1.375627        1.092858
APP_DESCRIPTION       APP_IMAGE      APP_DEVNUM      RATING_AVG      RATING_DEV         CUM_TOP
       1.371783        1.081724        1.022529        1.106210        1.236742        1.100730
```

```
> #Final Model
> finallog<- glm(APP_DEAD ~. , family=binomial(link="logit"), na.action=na.pass, data=mydata4, )
> summary(finallog)

Call:
glm(formula = APP_DEAD ~ ., family = binomial(link = "logit"),
    data = mydata4, na.action = na.pass)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3618  -0.9595  -0.5897   1.0120   2.4598

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      6.9052021  1.3337615   5.177 2.25e-07 ***
APP_ISPAID      -0.8896688  0.1150217  -7.735 1.04e-14 ***
INAPP_PURCHASE  -0.8925651  0.1153323  -7.739 1.00e-14 ***
TOP_DEVELOPER   -0.2267693  0.2844677  -0.797  0.42535
CONTENT_TYPE    -0.0520123  0.0515085  -1.010  0.31260
APP_ENGLISH     -1.0656039  0.4159358  -2.562  0.01041 *
INSTALL_SIZE     0.0004572  0.0004656   0.982  0.32607
APP_DESCRIPTION -0.0214123  0.0067919  -3.153  0.00162 **
APP_IMAGE       -0.0736729  0.0070020 -10.522  < 2e-16 ***
APP_DEVNUM      -0.0067199  0.0013009  -5.165 2.40e-07 ***
RATING_AVG      -0.4691008  0.2330047  -2.013  0.04409 *
RATING_DEV       0.8347211  0.1494865   5.584 2.35e-08 ***
CUM_TOP         -0.5032507  0.1147013  -4.387 1.15e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3530.0  on 2578  degrees of freedom
Residual deviance: 3119.4  on 2566  degrees of freedom
AIC: 3145.4

Number of Fisher Scoring iterations: 4
> #Confusion matrix
> confusion_matrix(finallog)
           Predicted 0 Predicted 1 Total
Actual 0          1069         391  1460
Actual 1           466         653  1119
Total             1535        1044  2579
```

```
> confusion.log <- confusion.log[1:2,1:2]
> prop.table(confusion.log)
          Predicted 0 Predicted 1
Actual 0    0.4145017   0.1516092
Actual 1    0.1806902   0.2531989
```

```
> ##Decision Tree
> #Tree with all variables
> tree <- rpart(APP_DEAD ~. , data=mydata4, cp=.001, method="class",  parms=list(split="gini"))
There were 50 or more warnings (use warnings() to see the first 50)
> printcp(tree)

Classification tree:
rpart(formula = APP_DEAD ~ ., data = mydata4, method = "class",
    parms = list(split = "gini"), cp = 0.001)

Variables actually used in tree construction:
 [1] APP_DESCRIPTION APP_DEVNUM      APP_IMAGE       APP_ISPAID      CONTENT_TYPE
 [6] CUM_TOP         INAPP_PURCHASE  INSTALL_SIZE    RATING_AVG      RATING_DEV

Root node error: 1119/2579 = 0.43389

n= 2579

          CP nsplit rel error  xerror     xstd
1  0.2126899      0   1.00000 1.00000 0.022492
2  0.0202562      1   0.78731 0.79982 0.021604
3  0.0196604      4   0.72654 0.76139 0.021346
4  0.0151921      5   0.70688 0.72386 0.021064
5  0.0128091      6   0.69169 0.71403 0.020986
6  0.0089366     10   0.63986 0.69169 0.020800
7  0.0071492     11   0.63092 0.67828 0.020682
8  0.0062556     14   0.60947 0.68811 0.020769
9  0.0049151     15   0.60322 0.68990 0.020784
10 0.0044683     17   0.59339 0.69794 0.020853
11 0.0040214     21   0.57551 0.69526 0.020830
12 0.0035746     23   0.56747 0.68007 0.020698
13 0.0031278     25   0.56032 0.68275 0.020722
14 0.0029789     27   0.55407 0.68722 0.020761
15 0.0026810     34   0.52458 0.68811 0.020769
16 0.0023831     41   0.50581 0.69169 0.020800
17 0.0022341     44   0.49866 0.69616 0.020838
18 0.0020107     49   0.48704 0.69705 0.020845
19 0.0017873     55   0.47185 0.69794 0.020853
20 0.0013405     68   0.44772 0.69794 0.020853
21 0.0011915     76   0.43700 0.69437 0.020822
22 0.0011171     81   0.42806 0.69616 0.020838
23 0.0010000     85   0.42359 0.69616 0.020838
> prune.t <- prune(tree, cp=0.0089366)
> predictions <- predict(prune.t, mydata4[,-13], type="class")
> confusion.dt <- table(predictions, mydata4$APP_DEAD)
> confusion.dt

predictions    0    1
          0 1082  338
          1  378  781
> prop.table(confusion.dt)

predictions         0         1
          0 0.4195425 0.1310585
          1 0.1465684 0.3028306
```

```
> #Tree with significant variables
> tree2 <- rpart(APP_DEAD~APP_ISPAID+INAPP_PURCHASE+APP_ENGLISH+APP_DESCRIPTION+APP_IMAGE+APP_DEVNUM+RATING_AVG+
+                    RATING_DEV+CUM_TOP, data=mydata4, cp=.001, method="class",  parms=list(split="gini"))
> printcp(tree2)

Classification tree:
rpart(formula = APP_DEAD ~ APP_ISPAID + INAPP_PURCHASE + APP_ENGLISH +
    APP_DESCRIPTION + APP_IMAGE + APP_DEVNUM + RATING_AVG + RATING_DEV +
    CUM_TOP, data = mydata4, method = "class", parms = list(split = "gini"),
    cp = 0.001)

Variables actually used in tree construction:
[1] APP_DESCRIPTION APP_DEVNUM      APP_IMAGE       APP_ISPAID      INAPP_PURCHASE
[6] RATING_AVG      RATING_DEV

Root node error: 1119/2579 = 0.43389

n= 2579

          CP nsplit rel error  xerror     xstd
1  0.2126899      0   1.00000 1.00000 0.022492
2  0.0160858      1   0.78731 0.80161 0.021615
3  0.0151921      5   0.71582 0.77748 0.021457
4  0.0128091      6   0.70063 0.76944 0.021402
5  0.0089366     10   0.64879 0.69526 0.020830
6  0.0071492     11   0.63986 0.67739 0.020674
7  0.0062556     12   0.63271 0.67650 0.020666
8  0.0058088     15   0.61394 0.67739 0.020674
9  0.0049151     17   0.60232 0.67024 0.020610
10 0.0031278     20   0.58624 0.66935 0.020602
11 0.0029789     30   0.55496 0.67382 0.020642
12 0.0029044     33   0.54602 0.67560 0.020658
13 0.0026810     38   0.53083 0.67560 0.020658
14 0.0020852     44   0.51475 0.67650 0.020666
15 0.0017873     49   0.50402 0.69616 0.020838
16 0.0015639     62   0.47811 0.69794 0.020853
17 0.0014894     68   0.46738 0.69884 0.020860
18 0.0013405     72   0.46113 0.69616 0.020838
19 0.0011915     80   0.45040 0.70241 0.020890
20 0.0010724     86   0.44325 0.70777 0.020935
21 0.0010000     91   0.43789 0.70777 0.020935
> prune.t2 <- prune(tree2, cp=0.0071492 )
> predictions2 <- predict(prune.t2, mydata4[,-13], type="class")
> confusion.dt2 <- table(predictions2, mydata4$APP_DEAD)
> confusion.dt2

predictions2    0    1
           0 1187  435
           1  273  684
> prop.table(confusion.dt2)

predictions2         0         1
           0 0.4602559 0.1686700
           1 0.1058550 0.2652191
```

```
> confusion.t.da <- table(mydata4$APP_DEAD,da$class)
> confusion.t.da

     0   1
  0 958 502
  1 351 768
> prop.table(confusion.t.da)

            0         1
  0 0.3714618 0.1946491
  1 0.1360993 0.2977898
```

```
> #Support Vector Machine
> mydata4$APP_DEAD = factor(mydata$APP_DEAD, level = c(0,1), label = c("Survival","Death"))
> model <- svm(APP_DEAD ~ ., data = mydata4)
> summary(model)

Call:
svm(formula = APP_DEAD ~ ., data = mydata4)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  1789

 ( 921 868 )


Number of Classes:  2

Levels:
 Survival Death
```

```
> pred = fitted(model)
> obs = ProjectData$APP_DEAD_CODE
> confusion.svm <- table(pred, obs)
> confusion.svm
          obs
pred       Death Survival
  Survival   352     1124
  Death      767      336
> prop.table(confusion.svm)
          obs
pred           Death  Survival
  Survival 0.1364870 0.4358278
  Death    0.2974021 0.1302831
```