Prediction on App Installation Through Data Mining

CS105

Guanying Deng
Zujing Zhao
Jingwen Guan

December 12th, 2018

Introduction

Cell phones play a crucial role in society now in a daily basis, and the use of apps is becoming popular. In this project, we are going to evaluate the popularity of Apps based on different features that each application contains. Our goal is to use data mining to build a model to predict number of installation for apps, based on rating, reviews, type, size and content rating. Classification learning will be used to analyze the dataset since the output attribute “Installs” is a single nominal output.

In this project, we begin by transforming the data between numeric and nominal. We then discretize some variables into different groups in order to make it easier for prediction. Also, SQLite will be used to organize numeric attributes into ranges, as well as finding minimum, maximum, and average values of each variables. The use of Python is data transformation, which transfer data between numeric and nominal attributes. Weka will split the dataset into training and testing data, and assist to build different prediction models by performing data mining algorithms. The result of data mining shows that reviews attribute has the highest accuracy in predicting installation. After performing R, we can see the relationship between each attributes and installation via visualization.

Data Description

The dataset collects data of 10 thousands Google Play Store apps for the Android market. The data includes information about category, rating, reviews, type(free or paid), price, content rating, genres, last update dates, current ver and android ver. The data set can be found on https://www.kaggle.com/lava18/google-play-store-apps.

Attributes	**Data Type**	**Values Represented**	**Description**
App	Nominal	“Coloring book moana”, “Sketch - Draw & Paint”, etc	The name of each App

Category	Nominal	ART_AND_DESIGN, BEAUTY, etc	The type of each App
Rating	Numeric	1 - 5	A number ranging from 1 to 5 that evaluates the App
Reviews	Numeric	1- 44893888	The total number of comments for each App from 1 to 44893888
Size	Nominal	1-100000	The size of the App from 1 to 100000(in k)
Type	Nominal	Free, Paid	Free if it's free to download the app
Price	Nominal	0, 4.99, 6.99, etc	How much it costs to download the App
Content Rating	Nominal	Everyone, Teen, Everyone 10+, Mature 17+, Adults only 18+, unrated	The required age ranges to download the App
Genres	Nominal	ART_AND_DESIGN, BEAUTY, etc	Same as Category
Last Updated	Nominal	2018/1/7, 2018/1/15, 2018/8/1, etc	The last updated date of each App, earliest May 21st, 2010 and latest Aug 8th, 2018
Current Ver	Nominal	1.0.0, 2.0.0, 1.2.4, etc	The version of each App
Android Ver	Nominal	4.0.3 and up, 4.2 and up, etc	The android version required to download each App

Installs (output)	Nominal	5000+, 10000+, 500000+, 100000+, etc.	The number of installation in ranges

Table1: Information of original database

The following 2 tables summarize the statistics obtained by SQL.

	MIN	**MAX**	**AVERAGE**
Rating	1	5	4.17355757
Reviews	1	44893888	294634.432
Size	1	100000	21088.5234

Table 3: Summary statistics 1

Attributes	**Categories**	**Counts**
Content Rating	Everyone	4942
	Teen	699
	Everyone 10+	242
	Mature 17+	297
	Adults Only 18+	2
Type	Free	5702
	Paid	480
	(0-3000]	2714

	(3000-750000]	1963
Installs (output)	750000+	3051

Table 4: Summary statistics 2

Data Preparation

Weka was used to remove the following attributes: app, category, genre, price, last updated, current ver and android ver.

App attribute was removed from the dataset because it is a unique identifier that causes overfitting. Category attribute was removed from the dataset because there are 32 possible categories in total. Since we think 32 is too large and might cause overfitting, we remove this input attribute. Genre attribute was removed from the dataset because it is a redundant attribute, that it contains the exact same information as the category attribute. Price attribute was removed from the dataset because it offers generally the same information as type attribute. In addition, the reason why we choose type attribute rather than price attribute is that there is a huge amount of possible prices for apps which might cause overfitting. Last updated, Android ver and current ver attributes were removed from the dataset because they are irrelevant, that this information don't help to predict the result.

Excel was used to delete commas by performing a simple find-and-replace. Since we stored database in CSV form, variables are separated by commas, and we have to use python later, so commas in examples have to be deleted in order for python to correctly write out a file.

Python was used to perform data transformation of size attribute. Since we're using k as the unit, for examples that end with "k", we delete "k", and for examples that end with "M", we replace "M" with "000" (1Mb=1000kb).   In this way, the nominal attribute is transformed into numeric attribute, in order to be meaningful in the later data mining processes.

Python was used to remove “+” in example’s install column. By removing “+”, we don’t mean to make install a numeric attribute. Since we have in total 19 possible results for install attribute, we want to group them into 5 groups to make it easier to analyze. Therefore, by removing “+”, we temporarily make install a numeric attribute, so we can later use Weka to discretize this attribute into 3 bins. In addition, we use python to divide installs into 3 groups: ‘1’ represents (-inf-3000], ‘2’ represents (3000-750000], ‘3’ represents (750000-inf). This discretization was obtained by Weka.

Python was also used to delete rows which contain unknown information (‘NaN’ in rating attribute, ‘Various with device’ in size attribute and ‘Unrated’ in Content rating attribute).

Weka was used to perform equal-height discretization of the following attributes into 3 groups: Rating, Reviews and Size. In addition, since there are too many possible outputs in Installs attribute (19 in total), we use Weka to discretize them into 3 groups.
After discretization, Rating attribute has 3 groups: (-inf-4.05], (4.05-4.45], (4.45-inf);   Reviews attribute has 3 groups: (-inf-285.5], (285.5-17068], (17068-inf); Size attribute has 3 groups: (-inf-8.55], (8.55-24500], (24500-inf); installs has 3 groups: (-inf-3000], (3000-750000], (750000-inf).

We used SQL to access minimum and maximum value of rating, reviews, size and price. We also used SQL queries to get the average value of each attributes.

Attributes	**Data Type**	**Values Represented**	**Description**
Rating	Numeric	1 - 5	A number ranging from 1 to 5 that evaluates the App

Reviews	Numeric	1- 44893888	The total number of comments for each App from 1 to 44893888
Size	Numeric	1-100000	The size of the App from 1 to 100000(in k)
Type	Nominal	Free, Paid	Free if it's free to download the app
Content Rating	Nominal	Everyone, Teen, Everyone 10+, Mature 17+ and Adults only 18+	The required age ranges to download the App
Installs (output)	Nominal	1->(-inf-3000], 2->(3000-750000], 3->(750000-inf)	The number of installation in ranges

Table5: Table of data after transformation and removing unnecessary attributes

Data Analysis

The primary goal is to explore the popularity of each Apps by observing how many installation has been placed with different attributes representative. In order to analysis the data, we first randomized the data to avoid any existed patterns in the data. After that we splitted the data to 80% for training and 20% for testing. We conducted OneR algorithm on the training data in order to get a baseline for later data modeling and analysis. OneR is a classification algorithm that computes one rule for each predictor in the data, and then selects the rule with the highest accuracy as the final rule. Also, in order to observe the highest accuracy, we tried algorithms like JRip, PART and J48 as well. JRip is an algorithm of decision tree application that predict the aiming variable of a new dataset. PART is an algorithm that generate a decision list builded up a partial C4.5 decision tree in individual iteration and predict the best outcome into the rule. J48 is a decision tree algorithm derived from each highest frequent outcomes

As we already deal with our raw data and discretized the numeric data in 3 groups, we simply used Weka to predict the highest accuracy through four different algorithms.   The best outcomes with highest accuracy derived from each algorithms are listed below on both training and testing data:

Algorithms	**Training**	**Testing**
JRip	85.3122 %	86.0194 %
PART	85.3931 %	86.4078 %
OneR	84.9563 %	85.89 %
J48	85.474 %	86.4078 %

Table6: Summary of Data-mining algorithms

Based on table3, 1R has the lowest accuracy in both Training and Testing data. The highest accuracy is taken place by J48 in both Training and Testing data(tie with PART), JRip and PART ranks in the middle. The average accuracy among these four algorithms is 85.2839% in training examples, and 86.1813% in testing examples. The standard deviation among the 4 algorithms is 0.00232 in training examples, and 0.00223 in testing examples. All four algorithms have very consistent prediction accuracies as all outcomes are around 85% in both training and testing data without significant fluctuation (standard deviations is very small, close to 0.002).

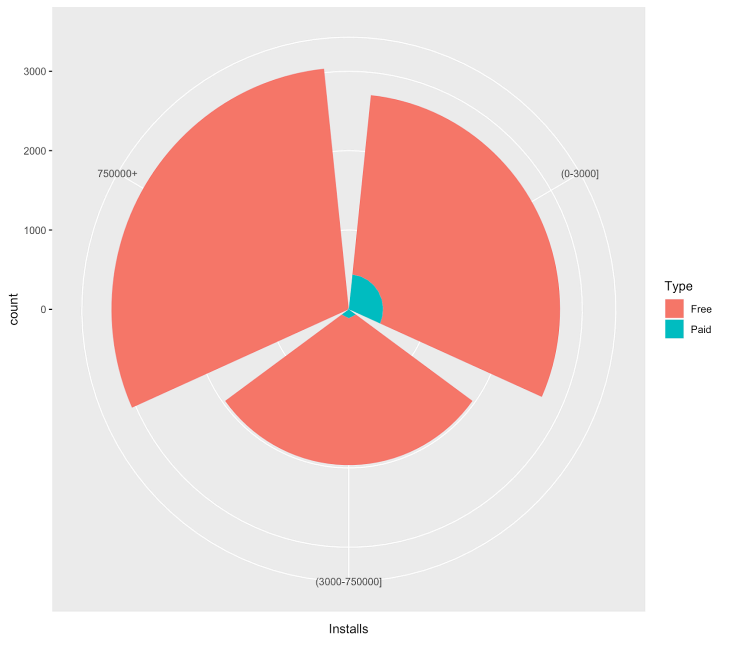
The training data helps to build up the proper model to predict highest possible outcome and the testing data is used to test whether the model can be considered as a good model.

Since we have both stable training and testing data, the model that we have to predict accuracy can be considered as good model and qualified to make any further prediction. J48 will definitely predict the best accuracy outcome, however, since all prediction of both training and testing from four algorithms are very closed, any algorithm can be picked for prediction if the data size is not significantly large. The result has minor difference. If the data size is significantly large, the output might result differently on either choose J48, or OneR.

Result

Based on the 4 data ming algorithms we performed in weka, training examples have 86.1813% (in average) accuracy on its model, and 85.2839%(in average) of testing examples are correctly predicted by the model obtained from training examples.

R was used to perform data visualization.



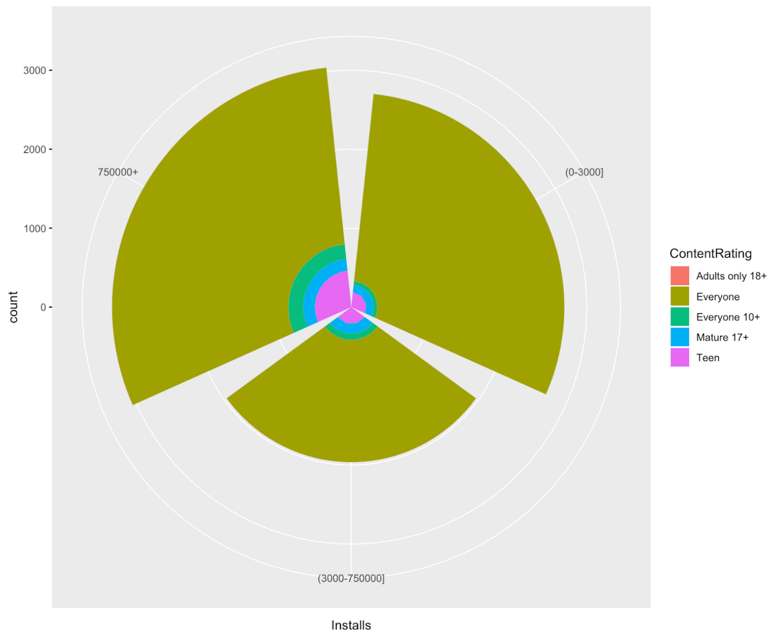


Diagram 1: Type v.s. Installs   *Diagram 2: Content Rating v.s. Installs*

Diagram 1 shows that most paid apps has installations less than 3000, and "paid" apps barely showed up in 750000+ plate.

Diagram 2 shows that most apps are rated "Everyone". Except for "Everyone", other apps are evenly split in the 3 installation groups.

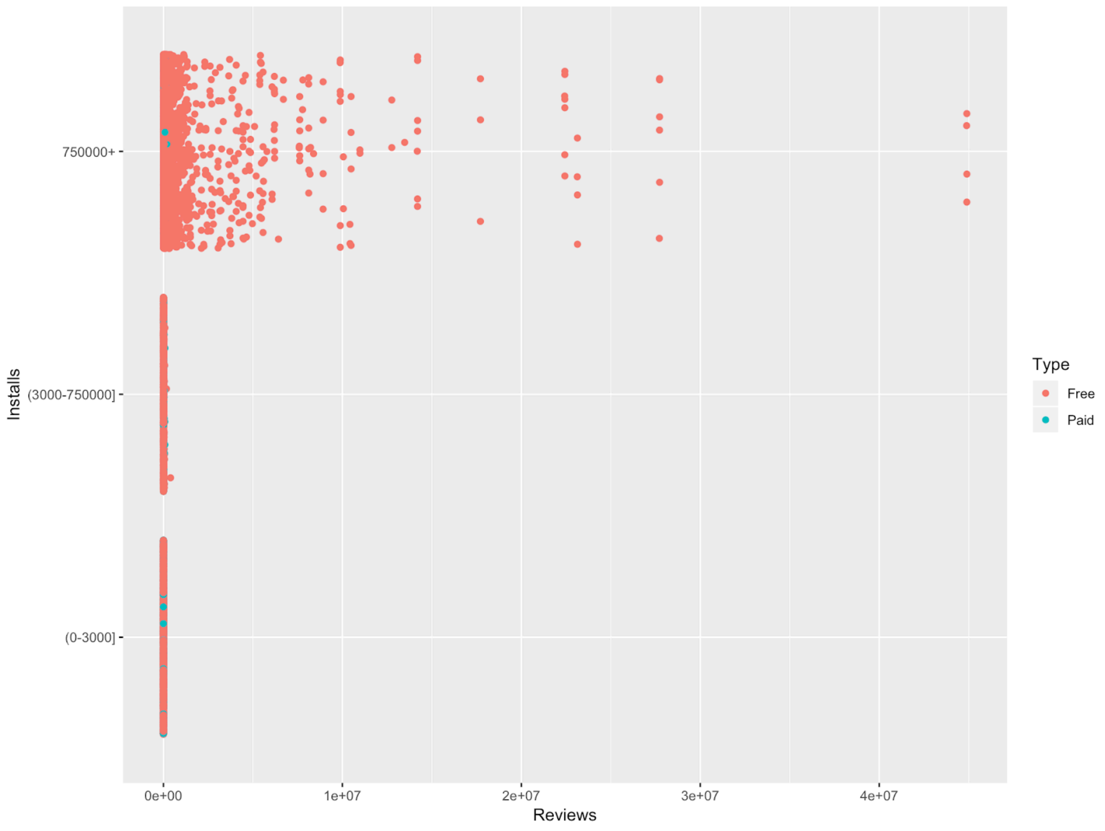


Diagram 3: Reviews v.s. Installs

Diagram 3 shows that apps with reviews greater than 200000(approximately) installation has installations greater than 750000. More reviews tend to be related with more installs. This is corresponded with the result we draw from 1R that reviews has the highest accuracy in predicting installations.

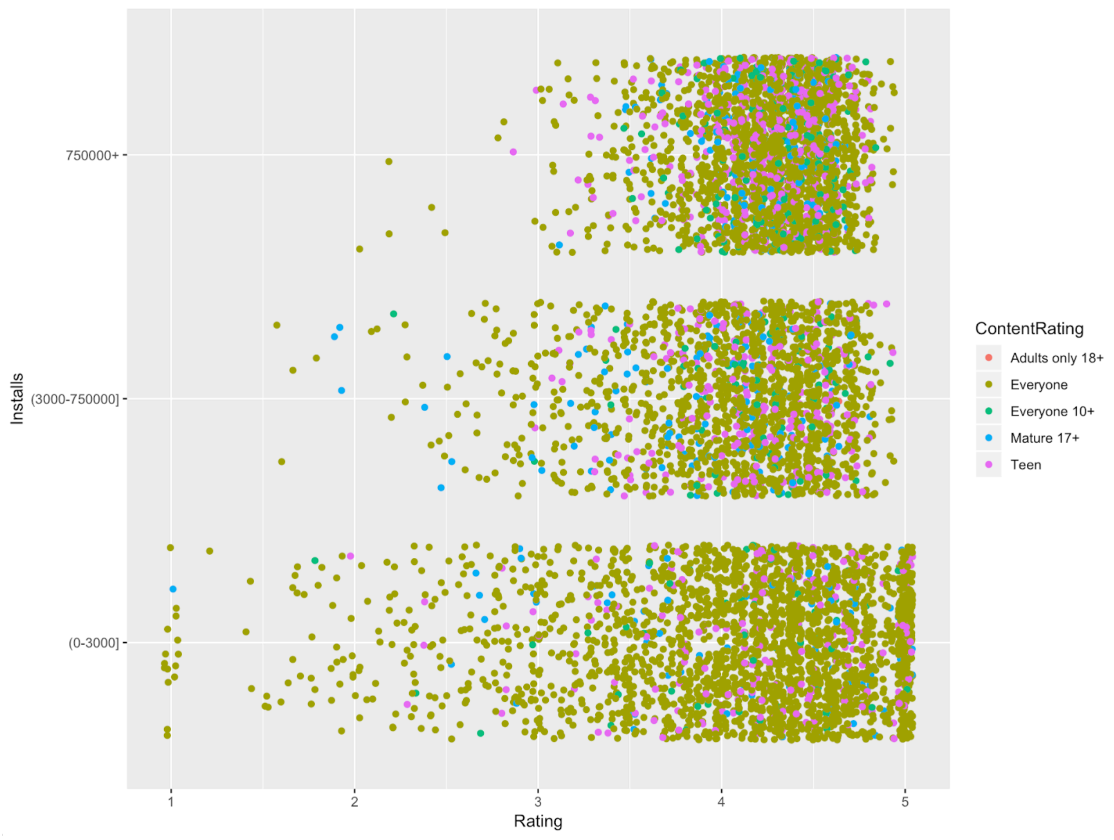


Diagram 4: Rating v.s. Installs

Diagram 4 shows that most apps with installation greater than 750000 has ratings greater than 4. It could be suggested from diagram 4 that installation tend to increase as rating of apps increases.

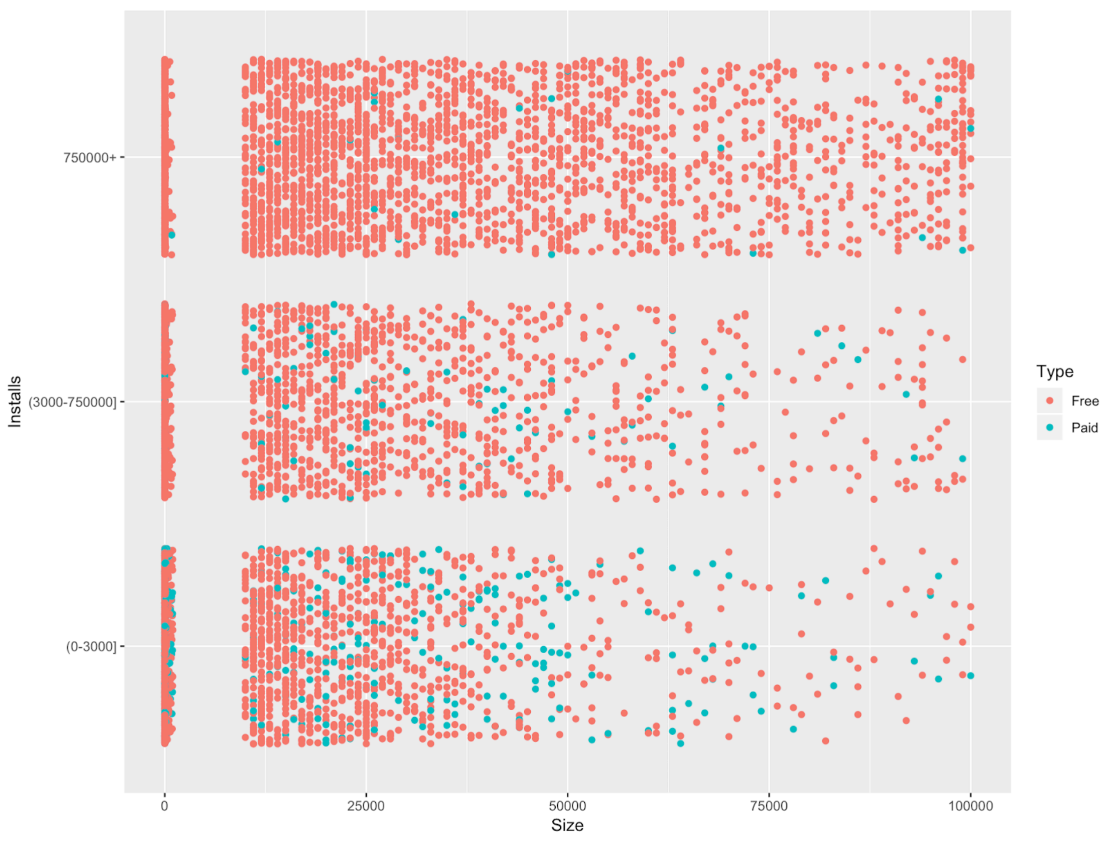


Diagram 5: Size v.s. Installs

Diagram 5 might suggest that apps with smaller sizes tend to have less installation, and as size of apps increases, installation may increase.

In addition, from this diagram we can see that most paid apps (blue points) are in the (0-3000] installation group, and paid apps with installations greater than 750000 are very rare.

We used two different algorithms and get two different confusion matrices.

Performance on 1545 test examples:

Model A:

		Predicted		
		(0-3000]	**(3000-750000]**	**750000+**
Actual	**(0-3000]**	470	64	0

	(3000-750000]	35	368	13
	750000+	0	106	489

Table7: Confusion Matrix based on 1R test examples

Overall accuracy=(470+368+489)/1545=85.89%

Error rate=1-85.89%=14.11%

Model B:

	Predicted			
Actual		**(0-3000]**	**(3000-750000]**	**750000+**
	(0-3000]	488	46	0
	(3000-750000]	46	361	9
	750000+	0	109	486

Table8: Confusion Matrix based on J48 test examples

Overall accuracy=(488+361+486)/1545=86.41%

Error rate=1-86.41%=13.59%

No significant differences were observed between the two confusion matrices. Model A is slightly better at classifying actual number of installation above 3000, and model B is slightly better at classifying actual number of installation below 3000.

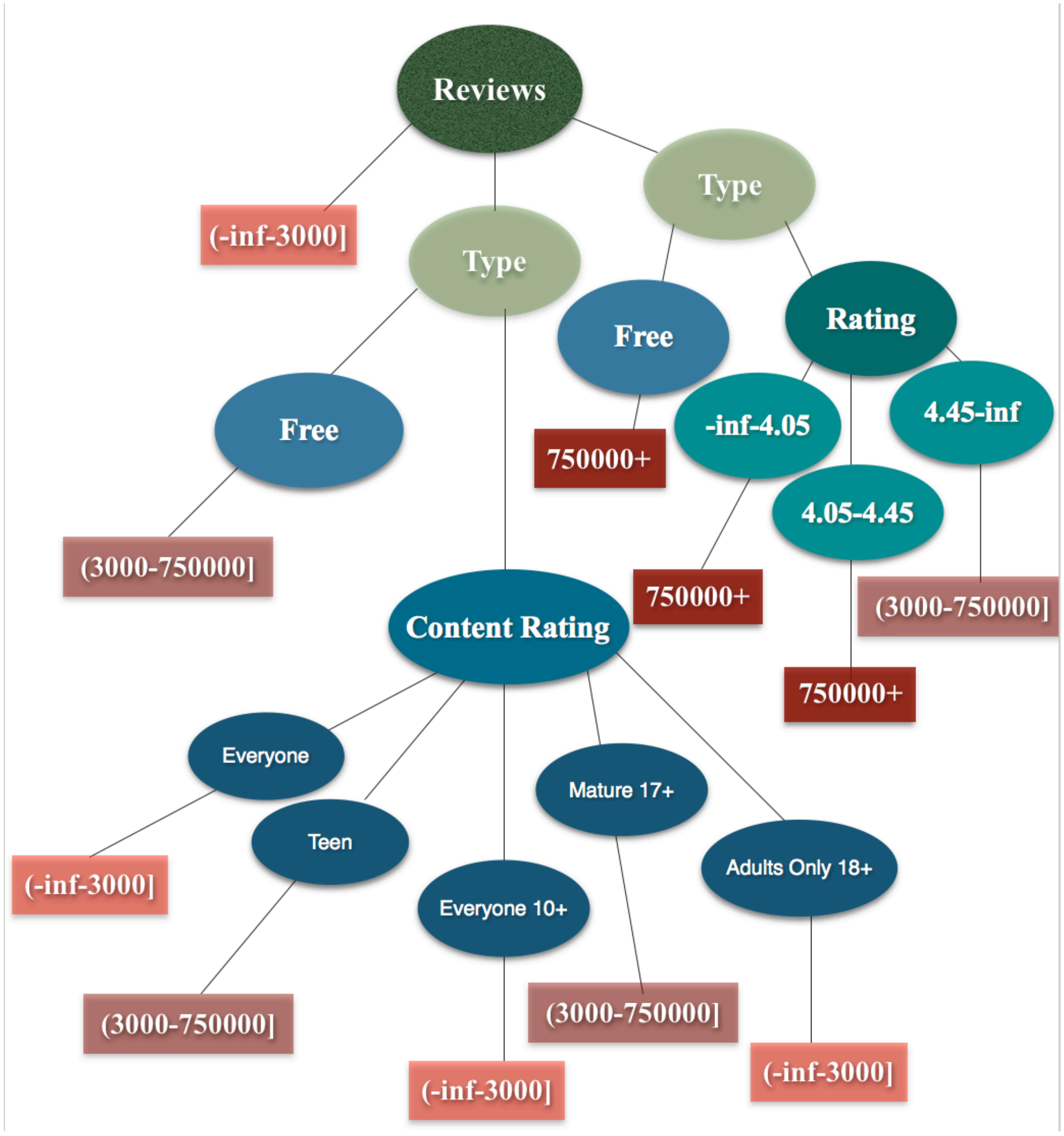
Conclusion

No prediction can be completely accurate, and the best we can do is to make the accuracy as high as possible. The 4 data mining algorithms we performed in weka suggests that the models obtained by training examples can give an accurate prediction of how many installations (in one of the three groups) the apps have about 86% of the time. Both of models obtained from OneR and J48 tree show that the

Reviews attribute may have the biggest influence on installation. The more reviews an app gets, the more people would install that app. This result may suggest that users treat reviews as a benchmark to determine whether to download an app. Thus, app designers could take advantage of this insight and attract installations by encouraging users to write reviews.

Appendix A: Rules/Tree

- **J48 tree:**



- **OneR rule:**

Reviews:

'(-inf-285.5]' -> 1

'(285.5-17068]' -> 2

'(17068-inf)' -> 3

(5252/6182 instances correct)

- **PART rules:**

Reviews = '(17068-inf)' AND

```
Type = Free: 3 (2028.0/58.0)

Reviews = '(-inf-285.5]': 1 (2070.0/135.0)

Type = Free: 2 (1884.0/621.0)

Reviews = '(17068-inf)' AND
Rating = '(4.45-inf)': 2 (24.0/5.0)

Reviews = '(285.5-17068]' AND
Content Rating = Everyone AND
Rating = '(4.45-inf)': 1 (52.0/19.0)

Reviews = '(285.5-17068]' AND
Content Rating = Everyone AND
Size = '(8.55-24500]': 1 (28.0/9.0)

Reviews = '(285.5-17068]' AND
Content Rating = Everyone AND
Size = '(-inf-8.55]': 1 (23.0/8.0)

Reviews = '(17068-inf)': 3 (22.0/9.0)

Content Rating = Everyone AND
Rating = '(4.05-4.45]': 2 (14.0/4.0)

Content Rating = Everyone 10+: 1 (14.0/6.0)

: 2 (23.0/9.0)
Number of Rules  :   11
```

- **JRIP rules:**

```
(Reviews = '(285.5-17068]') and (Type = Free) => Installs=2 (1884.0/621.0)
(Reviews = '(285.5-17068]') and (Size = '(24500-inf)') => Installs=2 (51.0/22.0)
(Reviews = '(-inf-285.5]') => Installs=1 (2070.0/135.0)
(Reviews = '(285.5-17068]') => Installs=1 (103.0/37.0)
 => Installs=3 (2074.0/86.0)

Number of Rules : 5
```

Appendix B: Data Modification

```
# removeUnnecessary.py
# Guanying Deng (dguany@bu.edu)
# Zujing Zhao (zujingkl@bu.edu)
# Jingwen Guan (jwguan@bu.edu)
# This program performs some modifications to dataset, and
# write the modified dataset to a new file.

# open the file
infile=open('googleplaystore.csv', 'r')
outfile=open('googleplaystore00.csv','w')

# use comma to split between attributes
for line in infile:
    line=line[:-1]
    fields=line.split(',')

    # transforms sizes ending with Mb to kb (also transform nominal to numeric)
    if fields[4][-1]=='M':
        fields[4]=fields[4][:-1]+'000'
    if fields[4][-1]=='k':
        fields[4]=fields[4][:-1]

    # removes "+" at the end of installs
    if fields[12][-1]=='+':
        fields[12]=fields[12][:-1]

    # divide installs into 3 groups
    if fields[12] == '':
        continue
    if int(fields[12])<30000:
        fields[12]='1'
    elif 30000<=int(fields[12])<750000:
        fields[12]='2'
    else:
        fields[12]='3'

    # removes data for rows with missing values and writes to new file
    if fields[2]!='NaN' and fields[4]!='Varies with device' and fields[7]!='Unrated:
        print(fields[0]+','+fields[1]+','+fields[2]+','+fields[3]+','+fields[4]+','+fields[5]+','+fields[6]+','+fields[7]+','+fields[8]+','+fields[9]+','+fields[10]+','+fields[11]+','+fields[12],file=outfile)

infile.close()
outfile.close()
```

Appendix C: SQL Queries

```
SELECT MIN(Installs), MAX(Installs), AVG(Installs)
FROM googleplaystore;

SELECT MIN(Rating), MAX(Rating), AVG(Rating)
FROM googleplaystore;

SELECT MIN(Reviews), MAX(Reviews), AVG(Reviews)
FROM googleplaystore;
```

```
SELECT MIN(Size), MAX(Size), AVG(Size)
FROM googleplaystore;

SELECT MIN(Price), MAX(Price), AVG(Price)
FROM googleplaystore;

SELECT count(*)
FROM googleplaystore
WHERE Content Rating = Everyone;

SELECT count(*)
FROM googleplaystore
WHERE Content Rating = Teen;

SELECT count(*)
FROM googleplaystore
WHERE Content Rating = Everyone 10+;

SELECT count(*)
FROM googleplaystore
WHERE Content Rating = Mature 17+;

SELECT count(*)
FROM googleplaystore
WHERE Content Rating = Adults Only 18+;

SELECT count(*)
FROM googleplaystore
WHERE Type = Free;

SELECT count(*)
FROM googleplaystore
WHERE Type = Paid;
```

Appendix D: Data Visualization Using R commands

```
> library(readxl)
> googleplaystore0022 <- read_excel("~/Desktop/CS105/Final
Project/googleplaystore0022.xls")
> View(googleplaystore0022)
> library(ggplot2)
> library(tidyverse)
> install.packages("tidyverse")
> ggplot(googleplaystore0022)+geom_bar(aes(x=Installs,
fill=ContentRating))+coord_polar()
> ggplot(googleplaystore0022)+geom_bar(aes(x=Installs, fill=Type))+coord_polar()
> ggplot(data=googleplaystore0022, mapping=aes(x=Reviews,
y=Installs))+geom_point(aes(color=Type), position="jitter")
> ggplot(data=googleplaystore0022, mapping=aes(x=Rating,
y=Installs))+geom_point(aes(color=ContentRating), position="jitter")
> ggplot(data=googleplaystore0022, mapping=aes(x=Size,
y=Installs))+geom_point(aes(color=Type), position="jitter")
```