BU 425 - Section A

Salar Ghamat

BU 425 Final Project

Click Fraud Detection

December 15, 2021

KW Analytics

Frederikke Noerager

Julie Ngo

Maggie Lin

Maitry Mistry

Rick Miao

Introduction

Business Problem and Motivation

This report analyzes a company's ability to detect click fraud through predictive models and exploratory analysis. The popularity of the internet has given rise to many new forms of fraud. Pay-per-click advertising (PPC), introduced in the late 1990s, is when companies pay a fixed amount for each person clicking on their online advertisement. This has been an important marketing tool for companies worldwide, but it has paved the way for click fraud. The first click frauds were detected in the early 2000s; since then, its volume has continued to grow exponentially (Clickcease, 2020).

PPC is an advertising model where advertisers pay publishers every time a user clicks an advertisement link on the publisher's page (CFI, 2021). Click fraud occurs when some person or software clicks on PPC ads, posing as legitimate customers without any interest in the ad's contents. Advertisers need to pay each time someone clicks their ad. In the case of click fraud, this becomes a wasteful expense as the person clicking has no real interest in the ad. The average cost of one click on a search engine, such as Google, is around $1, which may seem like a small and insignificant amount. However, with the speed of software and malicious bots, click fraud has cost advertisers $35 billion annually (Clickcease, 2020). Due to companies' dependence on PPC ads, they encompass a large portion of their marketing expenses. Consequently, click fraud poses a real threat to their survival. On average, 14% of clicks are invalid, which means 14% of their PPC expenses are wasted (Avital, 2020). Moreover, click fraud increases web traffic, further wasting the company's network resources and potentially blocking legitimate users. Additionally, click fraud builds a skewed image of an ad's performance, making it look more popular than it really is. These false impressions can be misleading for performance evaluations and predictions that often form the basis of important marketing and investment decisions. Therefore, the ability to detect click fraud is highly valued by companies looking to advertise via PPC.

TalkingData

China's largest big data service platform, TalkingData, "offers data collection, integration and operational analysis, and marketing and results-monitoring" (TalkingData). With these services, they "help enterprises analyze user behavior, mine user value, optimize advertising strategy, and improve marketing effectiveness" (TalkingData). As a result of their data collection, they deal with over 3 billion clicks per day (Kaggle, 2017). Due to the high volume of traffic, they encounter significant fraudulent activity. We aimed to develop analytical models so that TalkingData can proactively detect fraudulent users and block them.

Description of Data

We built predictive and exploratory models. Four predictive models (logistic regression, induction tree, neural network, and k-nearest neighbours) were used to determine whether a user will download an app after clicking an ad. The models were then assessed and compared to determine which one is the most capable of detecting click fraud. We conducted an exploratory analysis to cluster similar IP addresses and identify if there are any groups that appear fraudulent. This allowed us to create an IP blacklist for TalkingData to predict fraudulent attempts in the future.

The dataset on Kaggle provided by TalkingData includes 4 million rows, each corresponding to clicks, over the span of four days. However, due to technical limitations, we used a sample of 100,000 rows. The dataset had 8 columns:

Feature	**Description**
`ip`	The IP address of the click
`app`	The app ID for marketing
`device`	The device type ID of the users' mobile phone
`os`	Operation system ID of users' mobile phone

channel	The channel ID of the ad publisher
click_time	The timestamp of the click
attributed_time	The timestamp of the app downloaded if the app was downloaded after click the ad
is_attributed	Whether the app was downloaded or not

The predictive models predicted the is_attributed feature to determine whether a user would download the app using the other features as dependent variables. The clustering model used the features to cluster similar clicks to determine fraudulent IP addresses.

Overall Data Preparation

For the models, the attributed_time feature was removed because it was only present if the user downloaded the app. Since we sampled a portion of the dataset, the click_time attribute was removed as it is not an accurate reflection of when the clicks occurred. Lastly, for each predictive model, we divided the dataset into the train and test set, with 75% and 25% of the samples respectively.

Consolidated Dataset

We consolidated the original dataset using the fields ip, app, device, channel, os, and is_attributed. This effectively looks for users that are using the same ip, app, device, channel, and os combination to repeatedly click on an ad. The frequency (freq) was aggregated by adding up the number of rows with the given combination. The motivation behind this is that we wanted to observe the number of clicks that a particular IP address makes from a certain device using a specific OS made for each app within a channel, to hint at the IP addresses that were conducting click fraud. This is because click fraud is typically performed by clicking multiple times on a particular advertisement to make it appear as though there was a lot of engagement.

Smote Dataset

The original dataset was skewed such that only 0.227% of clicks had `is_attributed = 1` (meaning the user downloaded the app) and 99.773% of the clicks had `is_attributed = 0`. Since the data was unbalanced, we used the smote function to oversample the minority class and undersample the majority class. The smote function generated a new dataset of 199,987 clicks, of which 20,203 have `is_attributed = 1` and 179,784 have `is_attributed = 0` (approximately 10% of the dataset are positive). This balances the dataset while also ensuring that samples from the majority class still remain in high proportion.

Analysis

We aim to get low train and test error rates that are smaller than the respective base rates of the dataset. We also wanted small false positive rates and false negative rates. A false positive indicates we have predicted that the user will download when they in fact do not. A false negative indicates we have predicted that the user will not download when they in fact do download the app. A small false positive rate is important because we do not want to identify anyone as downloading when they are in fact not downloading and are potentially fraudulent. A small false negative rate is relevant because we do not want to mistakenly assume that someone does not download the app, as they could then be classified as fraudulent during the exploratory analysis when they are in fact not fraudulent. It is critical to have a small false negative rate because we do not want to mistakenly classify a user as fraudulent and block their IP address when they are in fact legitimate.

Predictive Analysis - *Linear Regression Model*

Original Dataset

To compute the linear regression model, we used the `step` function to determine that the independent variables `ip`, `app`, `channel`, `device`, and `os` should be used to predict the target variable `is_attributed`. For the dataset, the base error rates are 0.2187% and 0.252% respectively. The errors for the train and test set are 0.224% and 0.26%. The false negative rate is 98.41% and the false positive rate is 0.012%. This model's train and test errors are higher than the base error rates and the high false negative rate indicates that this model poorly fits the dataset.

Consolidated Dataset

For the consolidated dataset, the same features, along with the frequency, were used for the model. All of the features except frequency are significant, indicating that adding `freq` did not improve the model accuracy. The base error rates for the train and test set are 0.2315% and 0.2328%, respectively. The train error is 0.2356%, the test error is 0.2369%, the false negative rate is 100%, and the false positive rate is ~0%. Since the model error rates are not identical to the base rates and the FP rate is not exactly 0, the model did perform better than the base model. However, the FN rate is higher than the previous model, indicating that the adding `freq` did not improve the accuracy.

Smote Dataset

For the smote dataset, the base error rates for the train and test set are 10.05% and 10.25% respectively. The model train and test error rates are 9.01% and 9.13%. Since the train and test error rates are smaller than the base rate, the model is able to accurately predict the target variable. The FN rate is 78.64%, which means the model has improved from the previous model. The FP rate is low at 1.19%,

which means the model can accurately predict if the user will not download the app.

Refer to Exhibit 1 for the summary of the linear regression model.

Predictive Analysis - *Induction Tree Model*

Original Dataset

In addition to the modifications mentioned earlier, we also converted the `is_attributed`, `app`, `channel`, `device` and `os` fields into categorical variables. Using the `printcp` function, we find that the optimal complexity is `cp = 0.00867`. The base error rates are 0.228% and 0.224% for the train and test set respectively. The train and test errors are 0.136% and 0.216%. The false negative rate is 64.29% and the false positive rate is 0.072%. The FN rate is lower than the linear regression model, indicating that it is less likely to misclassify someone as fraudulent. Since the train and test errors are close to the base error rate, it means that the model may benefit from over or undersampling to account for the imbalance in the dataset.

Consolidated Dataset

For the consolidated dataset, the same features were used, scaled, and converted to factors as the previous model. The complexity was kept the same as previously, for comparison purposes. The base error rates are 0.238% and 0.212% for the train and test set respectively. The train and test errors are 0.136% and 0.245%. The FN rate is 78.85% and the FP rate is 0.078%. The model performs similarly to the original dataset as the error rates are close in value, but the false negative rate is much higher compared to the original dataset. Thus, consolidating the dataset did not improve the accuracy.

Smote Dataset

For the smote dataset, the base error rates for the train and test datasets are 10.17% and 9.89%. The model has a train error rate of 0.438% and test error rate of 4.96%. Although the test error rate is slightly higher than the train rate, the model is a good fit because both of the error rates are small in value and lower than their respective base rates. Additionally, the false negative rate of 47.52% is lower than the original dataset and the previous model, meaning it can most accurately determine if a click will lead to downloading the app. The false positive rate is higher than the original data set due to a higher proportion of samples with `is_attributed = 0`, but it is still low at 0.29% indicating a good fit.

Refer to Exhibit 2 for the summary of the induction tree models.

Predictive Analysis - *Neural Networks*

The neural network model used the `ip`, `app`, `device`, `os`, and `channel` fields to predict the target variable `is_attributed`. These fields were scaled between 0 and 1 so that no field would be weighed unfairly. Lastly, the `is_attributed` field was converted to a factor.

Original Dataset

A model was created with a complexity of 2 layers each with 4 and 2 nodes respectively. The base error rate for the train set is 0.24% and the test set is 0.188%. The train error rate of 0.24% and test error rate of 0.188%, are exactly the same as the base error rate. Additionally, the false positive rate of 0% and a false negative rate of 100% means that the model is classifying all inputs as `is_attributed = 0` since the data set has very few positive samples (227 out of 100,000 rows), and as such indicates that the model is performing at the base error rate level.

Another neural network model was created which increased the complexity from the previous

model to have 4 layers, each with 5 nodes to predict the target variable. This yielded the same results as the previous model, with the train and test errors identical to their respective base rates, a false positive rate of 0%, and a false negative rate of 100%. Thus, increasing the complexity did not improve the accuracy of the model.

Consolidated Dataset

For the consolidated dataset, we ran the more complex model with 4 layers and 5 nodes each. All of the features (including `freq`) were scaled, and the target variable is converted to a factor, similar to the previous model. The base train and test error rates are 0.234% and 0.225%. The train and test error rates are 0.234% and 0.225%. The FP rate is 0% and FN rate is 100%, meaning the model performed at the base rate, just like the original dataset, and did not improve the accuracy of the classification.

Smote Dataset

We applied the same model to the smote dataset. The base error rate is 10.10%. However, the model errors out as it was unable to converge. We attempted to increase the `threshold` level from 0.05 to 0.5, but the algorithm was still unable to converge. Due to technical limitations, we were unable to increase the `stepmax` field to allow for more steps.

Predictive Analysis - *K-Nearest Neighbours Model*

Original Dataset

The neural network model used the `ip`, `app`, `device`, `os`, and `channel` fields to predict the target variable `is_attributed`. The features were scaled by subtracting the minimum value and dividing by the range. Lastly, the `is_attributed` field was converted to a factor.

The k-NN model used Euclidean distance with 316 neighbours (`k = 316`) for prediction. This `k` value is optimal as it is approximately the square root of the sample size. The base rate for the train and test set is 0.21867% and 0.252% respectively. This model has a train error of 0.21867%, a test error of 0.0252%, a FP rate of 0%, and a FN rate of 100%. Thus, the model's train error is equivalent to its base rate and the test error rate is higher than its base rate, indicating poor model performance.

Consolidated Dataset

For the consolidated dataset, the features from the previous model were used along with the frequency of clicks, and the variables were scaled. The k-NN model was run with 316 neighbours. The base rate for the train set and test sets are 0.212% and 0.29% respectively. The train error is 0.212%, the test error is 0.29%, the FP rate is 0%, and the FN rate is 100%. Thus, the model performed at the base rate and simply classified all samples as `is_attributed = 0`. This shows that the model has low accuracy and does not perform better than the original dataset.

Smote Dataset

For the smote dataset, the k-NN model was run with 316 neighbours so that the results could be directly compared with the previous model. The base rate for the train set is 10.12% and for the test set is 10.05% . The model is accurate as the train error of 5.42% and test error of 5.31% are both small, similar in magnitude and lower than the base error rate. Moreover, the false negative rate is 39.35% which is much lower than any of the previous models and the false positive rate is also low at 1.45%.

Exploratory Analysis - *Clustering Model*

The k-means clustering algorithm was used for this analysis. We first extracted the `date` field from the `click_time` feature since the data is taken for clicks over a period of four days and was

consolidated based on the specific models below. Once the data was consolidated, we also removed the rows where `is_attributed = 1` since these indicate that the user downloaded the app after they clicked the ad, and as such is not fraudulent. Then, the remaining rows are potentially fraudulent attempts we will cluster. Since the other fields are categorical data, we focus on the frequency of clicks to perform clustering and assess whether the user is suspicious.

Consolidate by Number of Clicks per Date

For the first clustering model, the data was consolidated by the `ip`, `device`, `app`, `channel`, `os`, and `is_attributed` fields, and target variables were added that specified the frequency of clicks received on each of the four days (`Nov6`, `Nov7`, `Nov8`, `Nov9`). Next, we omitted the other features and only used the frequency of clicks variables for this model. By running the k-means algorithm for values of `k` ranging from 1 to 20 and calculating the respective R^2 values, we find that the optimal number of clusters is 6 (due to technical constraints, we were unable to run `fviz_nbclust` to graph the silhouette and total within sum of square values). This model with `k = 6` has a high R^2 value of 97.2%, indicating a good fit, as much of the variability in the dataset could be explained by the model. In cluster 2, there are an average of 3.89 clicks on Nov. 7, and in cluster 6 there are an average of 2.53 clicks on Nov. 8. The users in these clusters also had the highest average number of clicks each day compared to other clusters, which indicates that these clusters have users that are clicking more frequently than average and could potentially be fraudulent. See "Using Frequency of Each Date" under Exhibit 4 for the clustering model.

Consolidate by Number of Overall Clicks

For the second clustering model, the data was consolidated by counting the number of rows of the given `ip`, `app`, `device`, `channel`, and `os` combination in the whole dataset (which is over a fixed period of four days). We clustered on only the frequency variable. We similarly calculated the R^2 for

k-means for `k` ranging from 1 to 14, and obtained an optimal value of `k = 4`. This model has a high R^2 value of 93.9%, which is lower than the previous model, but it is still a good fit. In cluster 2, there is an average of 9.89 clicks across the four days. In cluster 4, there is an average 4.71 clicks across the four days. These high frequency numbers suggest that these users are clicking very frequently and therefore have a higher chance of being fraudulent. See "Using Overall Frequency" under Exhibit 4 for the clustering model.

Results

Predictive Model

There were four prediction models that were created and compared to find the best model that would accurately predict whether a device downloaded an app or not. Overall, it was decided that the induction tree model trained on the smote dataset would be the most accurate model for the company to use when predicting whether a click would lead to a download. This is because the test error rate was low and much smaller than the base rate compared to any other model. The false negative rate for this model was one of the lowest at 47.52%, compared to other models. Finally, the induction tree model was the only model where we were able to convert the features into categorical variables. Treating the features as factors and not numeric variables is essential because the features are not ordinal and as such there is little to no relationship between the values.

Exploratory Analysis

The clustering models showed us some groups of IP addresses which have high click frequencies but no downloads. As such, users that are in clusters with high frequency across either model may be suspicious and should be further investigated to confirm whether they are fraudulent. We found 91 IP addresses that may be suspicious out of the 34,857 distinct IP addresses in the dataset (see Exhibit 4 for

the list of flagged IP addresses). It should be noted that a hierarchical method could also be appropriate as we would not need to determine the number of clusters beforehand and it could also more clearly illustrate differences in frequency; however, we were unable to perform this due to technical limitations.

Evaluate

We were able to build an accurate induction tree model with the `ip`, `app`, `channel`, `device`, and `os` of the user to determine whether the user will download the app. Additionally, for the set of IP addresses that had not downloaded the app, we used exploratory analysis to analyze the frequency of clicks to determine if they were fraudulent to generate a list of suspicious IP addresses. We expected the artificial neural network to be a better predictive model as we can achieve higher model complexity but it was surprising to find that a classification model like an induction tree would be able to achieve lower error rates and false negative rates. To further improve the models for prediction and clustering, we could create more features with respect to the click frequency in smaller time periods (i.e. 5, 10, or 60 minute periods). This may result in more accurate predictions compared to the frequency per date and overall click frequency that was used in our current model.

Recommendation

We recommend that TalkingData use both the predictive and exploratory models to find which IP addresses are likely fraudulent to prevent click fraud. The predictive analysis, namely the induction tree, will allow them to find which IP addresses are likely to download the app once they have clicked on the ad. Then, the clustering will allow the company to find IP addresses that are likely to be fraudulent that they can further investigate to add to an exclusion list of IP addresses. In this way, TalkingData can reduce the amount of fraudulent traffic they receive and save significant costs and technical resources.

References

Avital, D. (2020) “Click Fraud 101:Common Types of Invalid Clicks and how they occur”. Cheq.ai (blog). Retrieved from: https://www.cheq.ai/click-fraud-101

CFI (2021), “Pay-Per-Click (PPC)”. corporatefinanceinstitute.com. Retrieved from: https://corporatefinanceinstitute.com/resources/knowledge/ecommerce-saas/pay-per-click-ppc/

Clickcease.com (2020) “A short history of ad click bots & PPC fraud”. Blogpost. Retrieved from: https://www.clickcease.com/blog/a-short-history-of-ad-click-bots-ppc-fraud/

Kaggle. (2017). “TalkingData AdTracking Fraud Detection Challenge”. Kaggle.com. Retrieved from: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/overview

TalkingData. (n.d.). Solutions. TalkingData. Retrieved from: https://www.talkingdata.com/solution.jsp?languagetype=en_us

Appendix

Exhibit 1: Logistic Regression Model

Original Dataset

```
> summary(Fit1)

Call:
glm(formula = is_attributed ~ app + device + channel + ip + os,
    family = "binomial", data = TrainSet)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0122  -0.0639  -0.0468  -0.0338   4.0804

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.726e+00  2.164e-01 -31.082  < 2e-16 ***
app          1.786e-02  1.712e-03  10.428  < 2e-16 ***
device      -1.297e-03  4.389e-04  -2.956  0.00312 **
channel     -4.665e-03  7.098e-04  -6.572 4.95e-11 ***
ip           1.066e-05  7.737e-07  13.771  < 2e-16 ***
os           2.425e-03  1.177e-03   2.061  0.03928 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2336.8  on 74999  degrees of freedom
Residual deviance: 2044.4  on 74994  degrees of freedom
AIC: 2056.4

Number of Fisher Scoring iterations: 9
```

Consolidated Dataset

```
> summary(Fit1)

Call:
glm(formula = is_attributed ~ app + device + channel + ip + os +
    freq, family = "binomial", data = TrainSet)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7945  -0.0664  -0.0493  -0.0370   4.0446

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  6.595e+00  3.819e+02   0.017  0.98622
app          1.673e-02  1.706e-03   9.806  < 2e-16 ***
device      -1.488e-03  5.579e-04  -2.667  0.00765 **
```

```
channel     -3.834e-03  6.705e-04  -5.718 1.08e-08 ***
ip           1.038e-05  7.638e-07  13.590  < 2e-16 ***
os           2.413e-03  1.082e-03   2.230  0.02572 *
freq        -1.336e+01  3.819e+02  -0.035  0.97208
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2402.9  on 73439  degrees of freedom
Residual deviance: 2121.5  on 73433  degrees of freedom
AIC: 2135.5

Number of Fisher Scoring iterations: 19
```

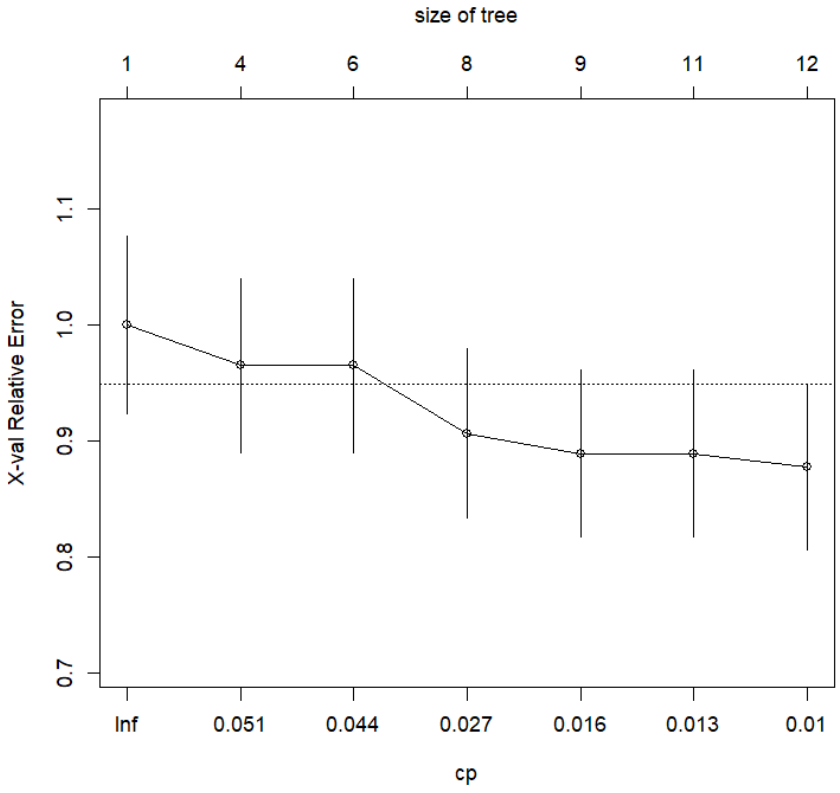
Smote Dataset

```
> summary(Fit1)

Call:
glm(formula = is_attributed ~ app + device + channel + ip + os,
    family = "binomial", data = TrainSet)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.5595  -0.4059  -0.2836  -0.1926   3.0934

Coefficients:
              Estimate Std. Error  z value Pr(>|z|)
(Intercept) -3.284e+00  2.759e-02 -119.029   <2e-16 ***
app          4.548e-02  6.003e-04   75.762   <2e-16 ***
device      -1.675e-03  9.588e-05  -17.472   <2e-16 ***
channel     -4.244e-03  8.375e-05  -50.671   <2e-16 ***
ip           1.045e-05  1.052e-07   99.260   <2e-16 ***
os          -5.573e-04  2.653e-04   -2.101   0.0357 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 97856  on 149989  degrees of freedom
Residual deviance: 74034  on 149984  degrees of freedom
AIC: 74046

Number of Fisher Scoring iterations: 6
```

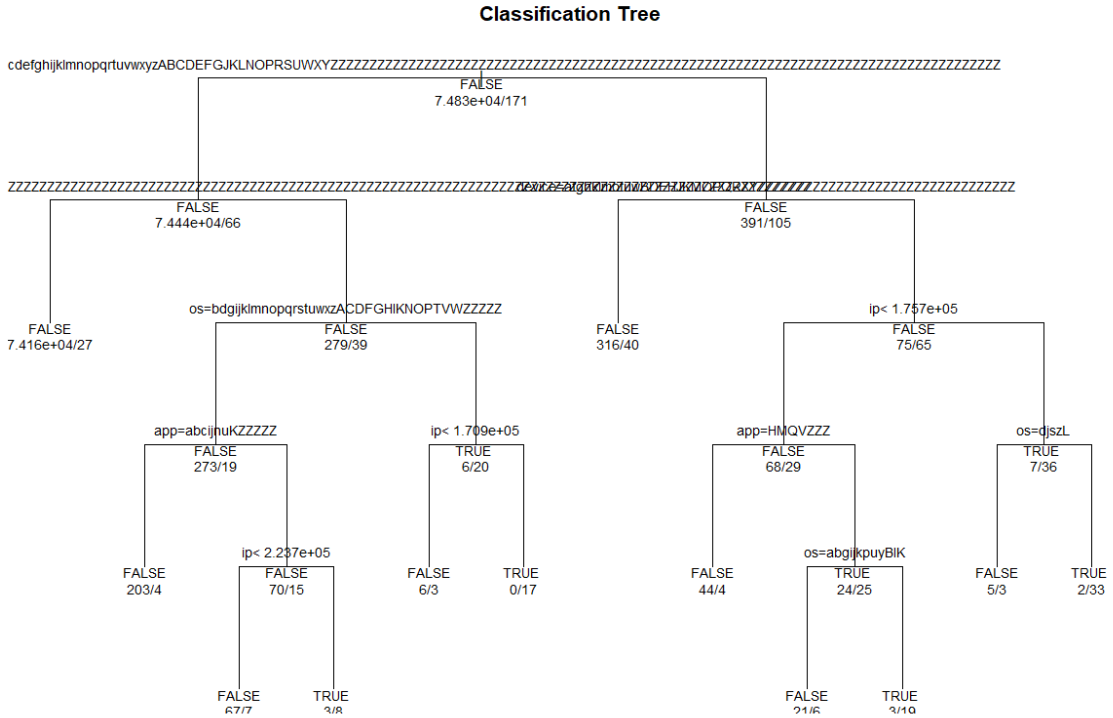
Exhibit 2: Induction Tree Model

Optimal Complexity

```
> printcp(fit1)

Classification tree:
rpart(formula = is_attributed ~ ip + channel + app + device +
    os, data = TrainSet, method = "class", cp = 0.00867)

Variables actually used in tree construction:
[1] app     channel device  ip      os

Root node error: 171/75000 = 0.00228

n= 75000

        CP nsplit rel error  xerror     xstd
1 0.056530      0   1.00000 1.00000 0.076385
2 0.046784      3   0.83041 0.96491 0.075036
3 0.040936      5   0.73684 0.96491 0.075036
4 0.017544      7   0.65497 0.90643 0.072731
5 0.014620      8   0.63743 0.88889 0.072025
6 0.011696     10   0.60819 0.88889 0.072025
7 0.008670     11   0.59649 0.87719 0.071551
```



Original Dataset



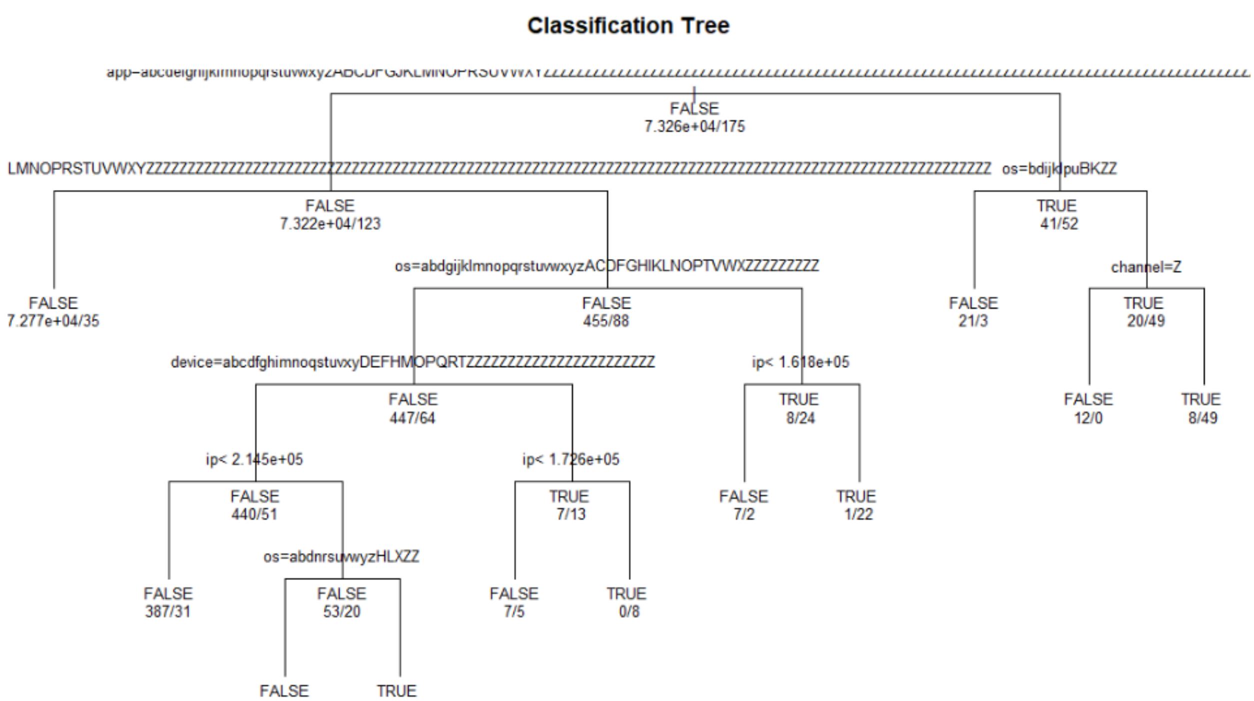

```
> print(fit1)
n= 75000

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 75000 171 FALSE (0.9977200000 0.0022800000)
   2)
app=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,20,21,22,23,24,25,26,27,28,29,30,31,32,33,36,
37,38,42,43,44,46,47,49,52,53,54,55,56,58,59,60,61,62,64,65,66,67,68,70,74,75,76,79,80,81,82,8
5,86,87,88,91,92,93,94,95,97,99,100,101,104,107,109,110,112,117,118,119,121,124,133,134,137,13
9,146,148,149,150,151,158,160,161,163,165,168,170,171,176,181,183,190,192,204,208,215,232,233,
266,268,271,273,310,315,363,372,394,398,407,486,536,538,548,551 74504  66 FALSE (0.9991141415
0.0008858585)
     4)
channel=3,4,13,15,17,18,19,21,22,24,30,101,105,107,108,110,111,115,116,118,120,121,122,123,124
,125,126,127,128,130,134,135,137,138,140,145,150,153,160,173,174,178,182,203,205,208,210,211,2
12,215,219,224,232,234,236,237,242,243,244,245,253,258,259,261,262,265,266,268,272,274,277,278
,280,282,315,317,319,320,322,325,326,328,330,332,334,340,341,343,347,349,353,356,360,361,364,3
71,373,376,377,379,386,391,400,401,402,404,406,409,410,412,416,417,420,421,424,430,435,439,442
,445,446,448,449,450,452,453,455,456,457,459,460,463,466,467,469,474,477,478,479,480,481,483,4
84,486,487,488,489,490,496,497 74186  27 FALSE (0.9996360499 0.0003639501) *
     5) channel=113,114,171,213,333 318  39 FALSE (0.8773584906 0.1226415094)
      10)
os=1,3,6,8,9,10,11,12,13,14,15,16,17,18,19,20,22,23,25,26,28,29,31,32,34,35,37,40,41,42,46,48,
49,53,55,58,132,866 292  19 FALSE (0.9349315068 0.0650684932)
        20) app=1,2,3,9,10,14,21,37,79,107,112,150,190 207   4 FALSE (0.9806763285
0.0193236715) *
        21) app=5,29,208 85  15 FALSE (0.8235294118 0.1764705882)
```

```
          42) ip< 223734 74    7 FALSE (0.9054054054 0.0945945946) *
          43) ip>=223734 11    3 TRUE (0.2727272727 0.7272727273) *
       11) os=4,7,27,30,36,43,47,61 26    6 TRUE (0.2307692308 0.7692307692)
         22) ip< 170852.5 9    3 FALSE (0.6666666667 0.3333333333) *
         23) ip>=170852.5 17    0 TRUE (0.0000000000 1.0000000000) *
    3) app=19,34,35,39,45,48,50,71,72,83,84,103,116,125,145,202 496 105 FALSE (0.7883064516
0.2116935484)
      6)
device=0,6,7,9,17,18,20,25,40,49,53,67,76,78,100,103,106,114,124,129,154,160,196,202,203,210,2
11,220,268,291,321,329,362,386,414,420,486,516,549,552,558,596,607,657,828,883,928,1080,1318,1
422,1482,1839,2980,3282,3545,3866,3867 356   40 FALSE (0.8876404494 0.1123595506) *
      7) device=1,4,16,21,30,33,50,56,60,74,97,102,109,116,180,188,579,957 140   65 FALSE
(0.5357142857 0.4642857143)
       14) ip< 175740 97   29 FALSE (0.7010309278 0.2989690722)
         28) app=34,39,45,50,72,103,202 48    4 FALSE (0.9166666667 0.0833333333) *
         29) app=19,35,48 49   24 TRUE (0.4897959184 0.5102040816)
           58) os=0,1,6,8,9,10,15,20,24,27,35,37 27    6 FALSE (0.7777777778 0.2222222222) *
           59) os=13,17,19,25,29,38,59 22    3 TRUE (0.1363636364 0.8636363636) *
       15) ip>=175740 43    7 TRUE (0.1627906977 0.8372093023)
         30) os=3,9,18,25,38 8    3 FALSE (0.6250000000 0.3750000000) *
         31) os=0,4,7,10,13,14,17,19,22,24,26,29,30,35,43 35    2 TRUE (0.0571428571
0.9428571429) *
```

Consolidated Dataset

```
> printcp(fit1)

Classification tree:
rpart(formula = is_attributed ~ ip + channel + app + device +
    os + freq, data = TrainSet, method = "class", cp = 0.00867)

Variables actually used in tree construction:
[1] app     channel device  ip      os

Root node error: 175/73440 = 0.0023829

n= 73440

        CP nsplit rel error  xerror     xstd
1 0.082857      0   1.00000 1.00000 0.075503
2 0.068571      2   0.83429 1.01143 0.075932
3 0.045714      3   0.76571 0.92000 0.072427
4 0.034286      5   0.67429 0.88571 0.071067
5 0.028571      6   0.64000 0.89143 0.071296
6 0.014286      7   0.61143 0.89143 0.071296
7 0.011429      9   0.58286 0.90286 0.071750
8 0.008670     10   0.57143 0.90286 0.071750
> plotcp(fit1)
> print(fit1)
n= 73440

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 73440 175 FALSE (0.9976171024 0.0023828976)
   2)
app=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,32,33,36,
37,38,39,42,43,44,46,47,49,50,52,53,54,55,56,58,59,60,61,62,64,65,66,67,68,70,74,75,76,80,81,8
2,83,84,85,86,87,88,91,93,94,95,99,101,103,104,105,107,109,110,117,118,119,121,122,124,125,133
,134,137,139,146,148,149,150,151,158,160,161,163,165,168,170,176,181,183,192,202,204,208,215,2
16,232,266,267,268,271,273,293,302,310,315,347,363,372,394,398,425,474,486,536,538,551 73347
123 FALSE (0.9983230398 0.0016769602)
     4)
channel=3,4,13,15,17,18,19,21,22,24,30,101,105,107,108,110,111,115,116,118,120,121,122,123,124
,125,126,127,128,130,134,135,137,138,140,145,150,153,160,173,174,178,182,203,205,208,210,211,2
12,215,219,224,232,234,236,237,242,243,244,245,253,258,259,261,262,265,266,268,272,274,277,278
```

```
,280,315,317,319,320,322,325,326,328,330,332,334,340,343,347,349,353,356,360,361,364,371,373,3
76,377,379,386,391,400,401,402,404,406,409,410,412,416,417,420,421,424,430,435,439,442,445,446
,448,449,450,451,452,453,455,456,457,459,460,463,466,467,469,474,477,478,479,480,481,483,484,4
86,487,488,489,490,496,497,498 72804  35 FALSE (0.9995192572 0.0004807428) *
     5) channel=113,114,171,213,282,333,341 543  88 FALSE (0.8379373849 0.1620626151)
      10)
os=0,1,3,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,28,29,31,32,34,35,37,38,40,4
1,42,46,48,49,50,53,55,56,58,76,132,135,607,866 511  64 FALSE (0.8747553816 0.1252446184)
        20)
device=0,1,2,4,6,7,9,11,20,21,25,33,37,40,49,50,56,58,76,78,79,100,114,124,129,154,160,167,203
,210,220,241,268,291,374,386,486,516,549,558,596,657,828,1042,1482,1728,2429,2980,3032,3282,38
66 491  51 FALSE (0.8961303462 0.1038696538)
          40) ip< 214518 418  31 FALSE (0.9258373206 0.0741626794) *
          41) ip>=214518 73  20 FALSE (0.7260273973 0.2739726027)
            82) os=0,1,3,13,17,18,20,21,22,24,25,34,38,50,58,76 56   9 FALSE (0.8392857143
0.1607142857) *
            83) os=6,10,15,19,29,32,37,866 17   6 TRUE (0.3529411765 0.6470588235) *
        21) device=16,30,60,74,97,109,180,188,579,581 20   7 TRUE (0.3500000000 0.6500000000)
          42) ip< 172611.5 12   5 FALSE (0.5833333333 0.4166666667) *
          43) ip>=172611.5 8   0 TRUE (0.0000000000 1.0000000000) *
      11) os=4,7,27,30,36,43,47,59,61,748 32   8 TRUE (0.2500000000 0.7500000000)
        22) ip< 161813.5 9   2 FALSE (0.7777777778 0.2222222222) *
        23) ip>=161813.5 23   1 TRUE (0.0434782609 0.9565217391) *
   3) app=34,35,45,48,71,72,96,108,115,116,145,261 93  41 TRUE (0.4408602151 0.5591397849)
     6) os=1,3,8,9,10,11,15,20,27,37,53,866 24   3 FALSE (0.8750000000 0.1250000000) *
     7) os=0,2,4,7,13,14,17,18,19,22,23,25,26,30,35,43,47 69  20 TRUE (0.2898550725
0.7101449275)
      14) channel=411 12   0 FALSE (1.0000000000 0.0000000000) *
      15) channel=3,5,21,101,203,213,243,274,320,419,465 57   8 TRUE (0.1403508772
0.8596491228) *
```



Smote Dataset

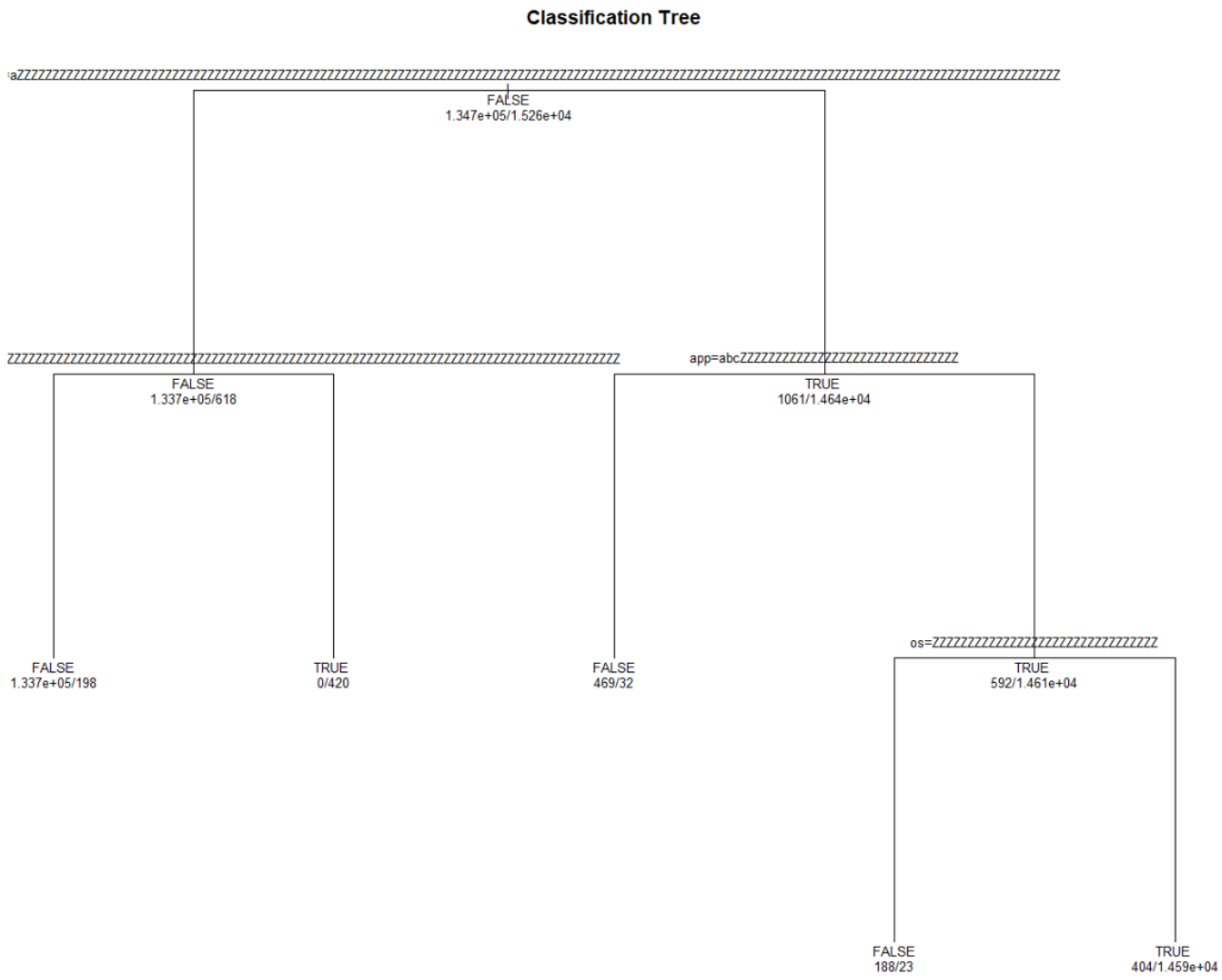
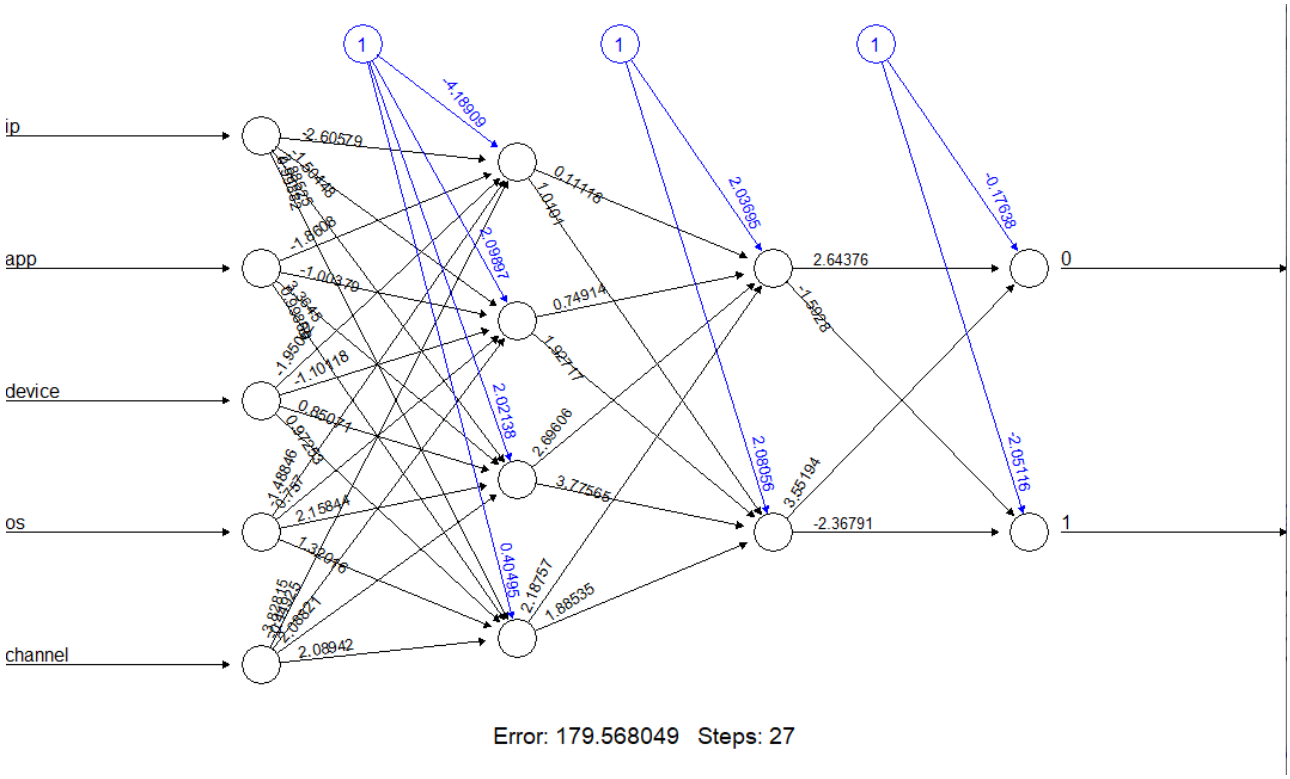
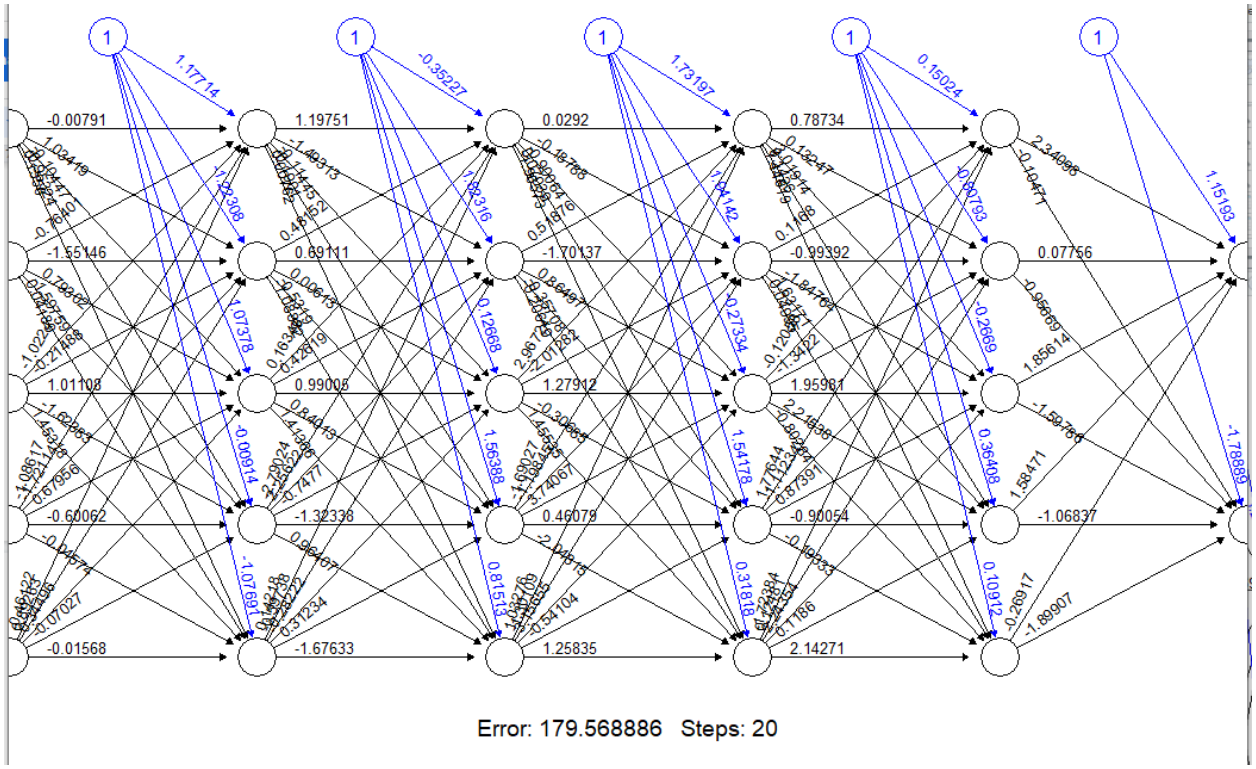


Exhibit 3: ANN Model

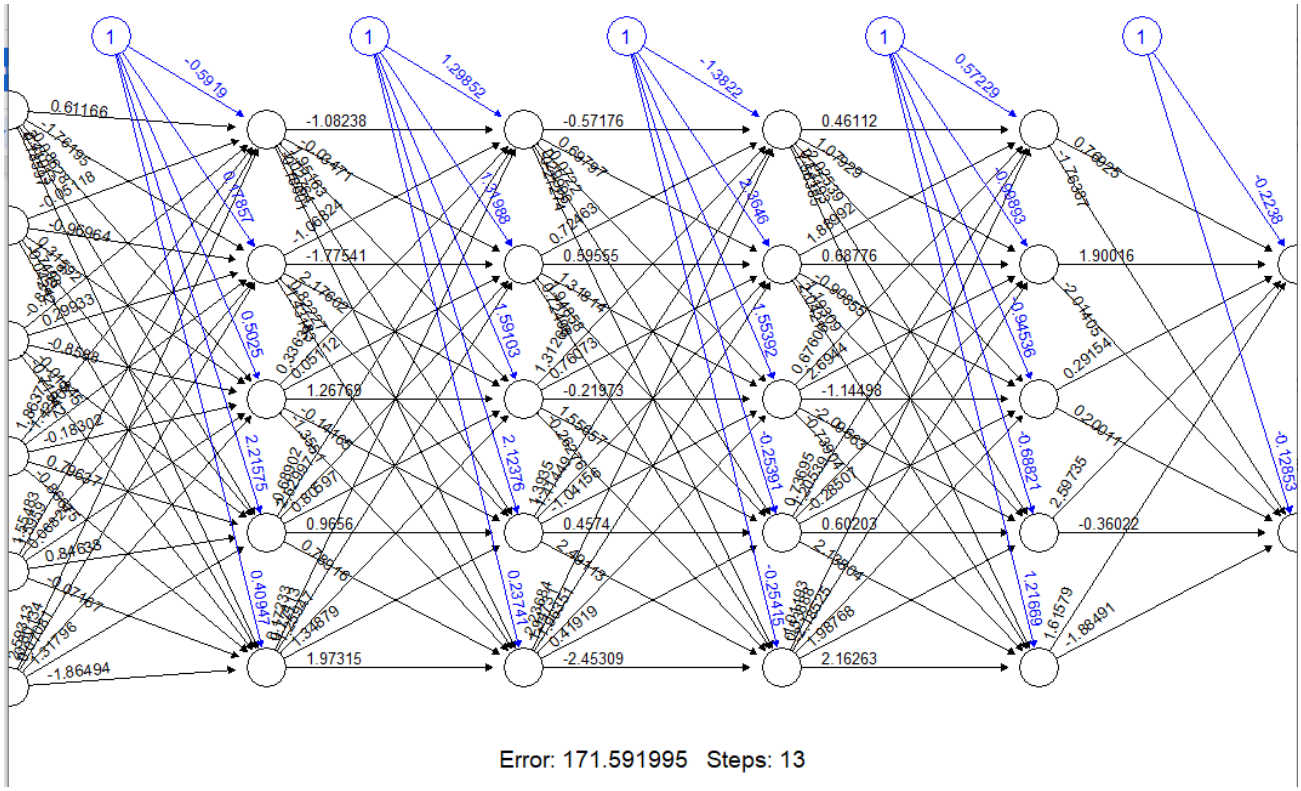
Original Dataset - Model has 2 layers with 4 and 2 nodes respectively



Original Dataset - Model has 4 layers with 5 nodes each



Consolidated Dataset - Model has 4 layers with 5 nodes each



Smote Dataset - Model has 4 layers with 5 nodes each

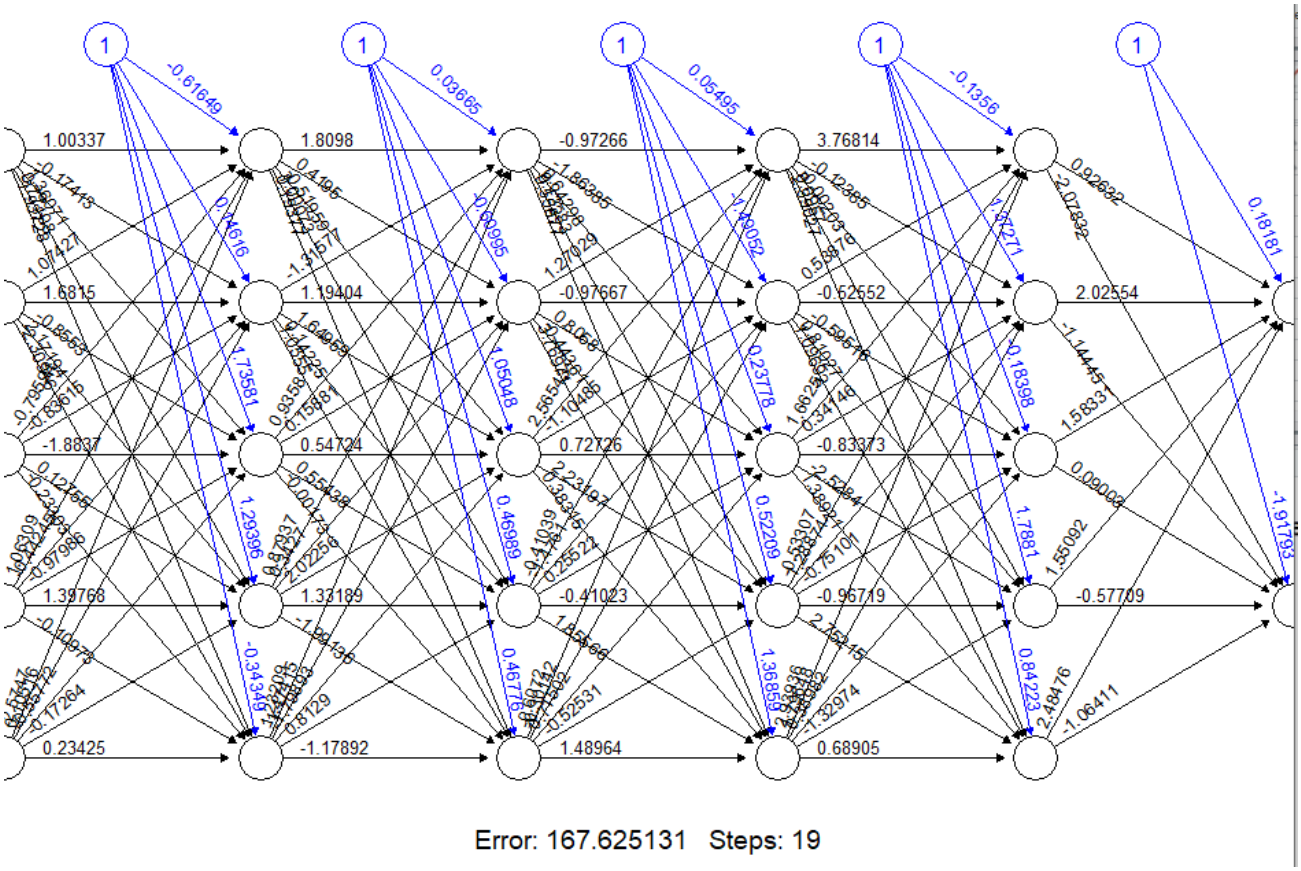
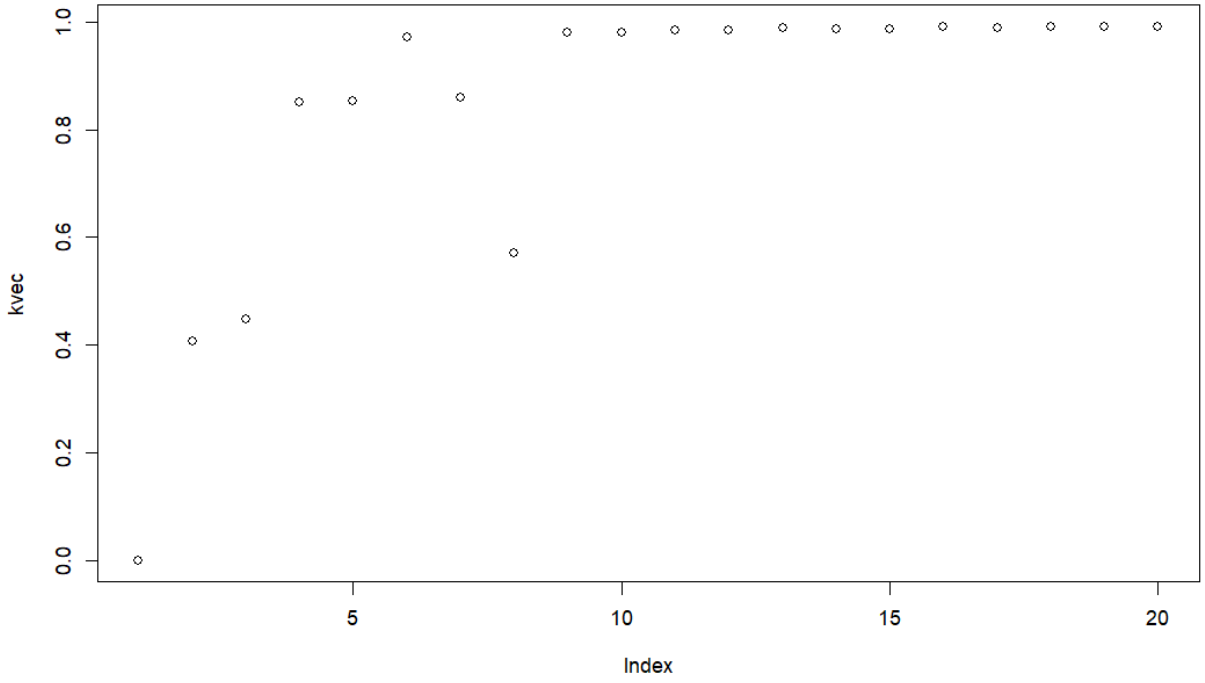


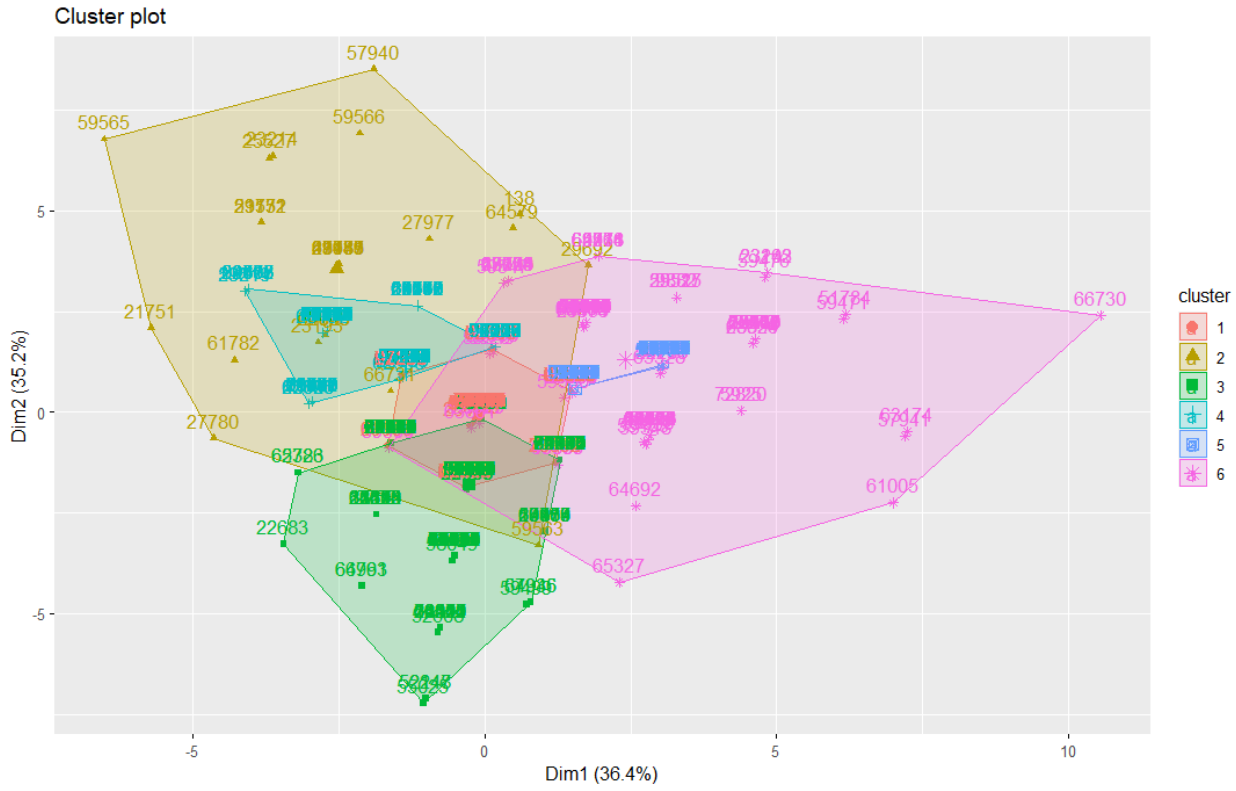
Exhibit 4: Clustering Model

Using Frequency of Each Date

R^2 plotted for k from 1 to 20



Cluster model




```
> print(cls6)
K-means clustering with 6 clusters of sizes 4949, 28, 28150, 31573, 32874, 119

Cluster means:
          Nov6        Nov7        Nov8         Nov9
1 1.004445e+00 0.012123661 0.009092746 0.0054556476
2 2.857143e-01 3.892857143 1.892857143 0.8928571429
3 3.552398e-05 0.007708703 0.008845471 1.0069271758
4 2.850537e-04 1.007601432 0.008298229 0.0006967979
5 6.083835e-05 0.000000000 1.005110422 0.0000000000
6 1.008403e-01 0.983193277 2.529411765 0.6974789916

Clustering vector:
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
   3    3    3    5    5    4    3    5    3    5    5    3    5    4    4    3    3
  18   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
   5    3    3    5    3    4    3    5    5    3    1    1    3    3    3    3    5
  35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51
   5    5    5    3    3    4    3    4    4    5    5    5    5    4    5    3    3
  52   53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68
   3    1    3    3    5    4    5    5    3    4    4    5    5    4    3    4    4
  69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85
   4    5    5    3    3    4    4    3    4    5    5    1    5    3    5    5    4
  86   87   88   89   90   91   92   93   94   95   96   97   98   99  100  101  102
   3    4    3    5    5    3    4    4    3    5    3    3    4    3    5    3    3
 103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119
   3    3    5    3    4    5    5    3    5    3    3    3    3    3    3    3    4
 120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136
```

```
  4    1    3    3    4    3    5    5    5    3    3    4    4    5    5    5    5
137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153
  4    2    3    4    4    3    4    5    4    3    4    3    5    4    3    4    4
154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170
  5    5    3    3    4    4    4    5    1    5    3    4    3    3    4    4    5
171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187
  5    3    4    3    5    5    4    5    5    5    4    3    3    3    3    5    5
188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204
  3    4    3    3    3    5    5    3    3    3    3    3    3    3    3    5    5
205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221
  3    3    3    3    5    5    3    3    4    5    4    3    3    4    4    3    5
222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238
  5    3    4    3    5    5    4    4    5    3    5    3    5    5    5    3    4
239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255
  4    5    3    4    3    4    5    4    5    3    3    5    5    3    4    5    4
256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272
  3    3    5    5    4    3    5    3    5    5    5    4    3    1    3    4    3
273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289
  4    3    5    3    4    5    5    3    3    3    4    5    3    4    5    4    5
290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306
  3    5    3    4    4    3    3    3    3    5    3    4    5    5    5    4    3
307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323
  4    4    1    1    1    3    3    4    4    3    3    1    4    4    3    4    3
324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340
  4    4    5    5    5    5    5    5    5    3    3    3    3    5    3    3    3
341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357
  4    4    5    4    4    3    5    4    3    5    5    3    5    5    3    4    5
358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374
  5    4    4    5    4    3    3    4    3    5    4    5    5    4    5    4    5
375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391
  4    3    4    4    3    5    5    3    4    3    5    5    5    5    3    3    3
392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408
  5    4    5    5    4    4    4    4    4    1    4    3    3    5    5    4    4
409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425
  4    1    5    4    5    4    4    3    5    1    1    4    4    4    3    4    3
426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442
  5    5    4    5    5    4    5    4    5    5    5    1    4    1    5    4    1
443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459
  1    3    5    1    3    1    4    5    5    3    4    3    4    4    5    3    5
460  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476
  3    5    5    4    4    4    4    1    4    4    3    4    4    4    3    5    3
477  478  479  480  481  482  483  484  485  486  487  488  489  490  491  492  493
  4    5    3    4    3    3    4    5    5    4    4    3    3    5    3    3    5
494  495  496  497  498  499  500  501  502  503  504  505  506  507  508  509  510
  5    3    5    5    5    5    1    3    3    4    3    3    5    3    5    4    4
511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527
  4    5    5    3    4    4    1    4    1    3    3    4    5    3    4    3    4
528  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544
  3    5    4    3    3    1    1    3    5    3    4    3    5    3    5    5    5
545  546  547  548  549  550  551  552  553  554  555  556  557  558  559  560  561
  4    4    5    5    5    4    3    3    3    3    3    3    4    1    5    4    3
562  563  564  565  566  567  568  569  570  571  572  573  574  575  576  577  578
  5    3    3    4    3    4    4    3    4    5    4    5    4    3    5    5    4
579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595
  5    3    5    4    5    5    4    5    5    3    4    3    5    4    3    3    3
596  597  598  599  600  601  602  603  604  605  606  607  608  609  610  611  612
  5    4    3    3    5    3    3    3    5    4    5    5    3    5    4    4    3
613  614  615  616  617  618  619  620  621  622  623  624  625  626  627  628  629
  3    3    3    5    5    5    3    3    5    4    3    5    4    5    4    5    5
630  631  632  633  634  635  636  637  638  639  640  641  642  643  644  645  646
  4    4    4    4    5    1    4    5    3    5    5    5    3    5    3    4    5
647  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663
  4    4    3    3    3    5    3    3    4    3    1    5    5    3    5    3    5
664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679  680
  3    5    5    5    5    5    3    3    5    5    3    3    4    3    5    4    4
681  682  683  684  685  686  687  688  689  690  691  692  693  694  695  696  697
  4    4    4    5    3    3    3    3    3    5    5    5    3    3    5    5    5
698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713  714
```

```
   5    4    5    4    4    5    3    4    3    4    5    5    4    5    3    5    4
 715  716  717  718  719  720  721  722  723  724  725  726  727  728  729  730  731
   4    5    4    4    5    3    5    5    4    5    4    5    4    4    5    5    3
 732  733  734  735  736  737  738  739  740  741  742  743  744  745  746  747  748
   5    5    4    3    4    1    3    1    5    5    4    3    5    3    5    3    1
 749  750  751  752  753  754  755  756  757  758  759  760  761  762  763  764  765
   1    4    5    4    4    5    4    4    5    4    5    3    3    5    4    5    5
 766  767  768  769  770  771  772  773  774  775  776  777  778  779  780  781  782
   4    4    3    4    3    5    4    3    3    1    3    1    4    4    5    4    4
 783  784  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799
   5    4    3    4    3    5    3    5    5    4    4    4    3    3    5    1    3
 800  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
   5    4    3    3    4    4    3    5    1    3    4    5    3    5    5    4    5
 817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833
   3    5    4    5    4    1    5    4    5    3    1    5    3    5    5    5    5
 834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849  850
   5    5    3    4    3    3    4    5    3    5    3    3    4    3    5    4    4
 851  852  853  854  855  856  857  858  859  860  861  862  863  864  865  866  867
   4    4    5    5    5    3    5    5    3    5    3    3    1    3    3    3    5
 868  869  870  871  872  873  874  875  876  877  878  879  880  881  882  883  884
   5    5    4    3    4    3    5    5    5    3    3    3    3    5    5    3    4
 885  886  887  888  889  890  891  892  893  894  895  896  897  898  899  900  901
   3    4    5    3    5    4    3    4    5    5    5    3    3    5    5    5    5
 902  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918
   5    4    3    4    5    5    5    5    3    3    3    4    5    5    5    5    3
 919  920  921  922  923  924  925  926  927  928  929  930  931  932  933  934  935
   5    3    5    3    3    5    4    5    3    4    4    5    4    4    3    3    3
 936  937  938  939  940  941  942  943  944  945  946  947  948  949  950  951  952
   3    5    5    5    3    3    3    3    5    3    4    5    3    4    4    4    3
 953  954  955  956  957  958  959  960  961  962  963  964  965  966  967  968  969
   4    5    5    1    3    5    5    5    5    3    4    3    4    5    4    1    4
 970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986
   4    5    3    3    3    4    4    5    4    4    5    4    3    5    5    1    5
 987  988  989  990  991  992  993  994  995  996  997  998  999 1000
   5    5    4    3    3    4    4    4    4    5    5    5    5    5
 [ reached getOption("max.print") -- omitted 96693 entries ]

Within cluster sum of squares by cluster:
[1] 154.6183 151.7500 732.7739 544.9836 169.1413 275.5126
 (between_SS / total_SS =  97.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

IP addresses in cluster 2 and cluster 6

```
> # cluster 2 rows
> newConsolidatedUnique[c(138,21751,21752,23135,23172,23185,23193,23214,25527,27780,27885,
+                         27977,28957,29692,49443,49776,51489,52609,57940,59531,59563,59565,
+                         59566,61782,64579,64943,66731,67681),]
          ip device app channel os is_attributed Nov6 Nov7 Nov8 Nov9
138     5314      0 151     347  0             0    0    3    3    0
21751  73487      1   3     153 13             0    0    6    2    3
21752  73516      1   3     153 13             0    0    4    1    0
23135  25679      1   2     205 13             0    0    3    1    0
23172  75634      1   2     205 13             0    0    4    1    0
23185 105433      1   2     205 13             0    0    3    1    0
23193 108341      1   2     205 13             0    1    4    2    2
23214 175837      1   2     205 13             0    0    5    2    0
25527   5348      1  15     245 13             0    1    5    2    0
27780   5314      1   3     280 13             0    1    4    1    3
27885  16741      1   3     280 13             0    0    3    1    1
27977  26995      1   3     280 13             0    0    3    2    0
28957 137052      1   3     280 13             0    0    3    1    0
```

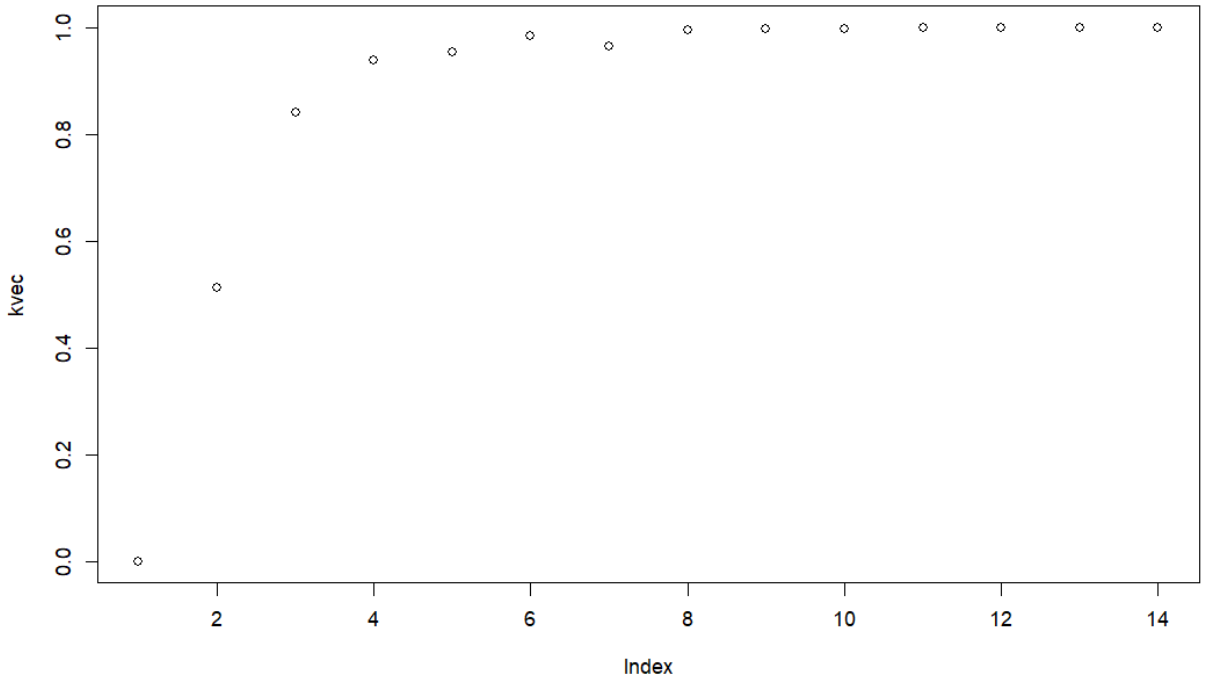
```
29692  73516      1  12     326 13             0    2    3    4    1
49443  73487      1  12     259 18             0    0    3    1    0
49776   5314      1   3     280 18             0    0    3    1    0
51489   5314      1  13     477 18             0    0    3    1    0
52609   5348      1  18     107 19             0    0    3    1    1
57940  73487      1   3     153 19             0    0    7    5    1
59531 105456      1   2     205 19             0    0    4    1    0
59563 175837      1   2     205 19             0    0    3    4    5
59565 178873      1   2     205 19             0    0    6    1    0
59566 209663      1   2     205 19             0    1    5    3    0
61782 114276      1   9     244 19             0    0    3    0    1
64579   5348      1   3     280 19             0    0    6    6    3
64943  40631      1   3     280 19             0    0    3    1    1
66731  73516      1  12     326 19             0    2    4    3    3
67681  55910      1   2     364 19             0    0    3    1    0


> # cluster 6 rows
> newConsolidatedUnique[c(5925,5926,5935,5973,9672,9882,12293,16787,16992,17177,17609,17883,
+
17941,18113,23125,23143,23145,23171,23188,23189,23197,23202,23215,23262,24513,24769,25328,2533
6,
+
25525,25794,25877,26376,26377,27050,27235,27518,27781,27978,28201,28641,28758,28801,
+
28836,28876,28976,29099,30486,34211,34589,34817,43721,47514,49777,51784,51862,52765,
+
52798,52815,55170,57941,58205,58745,59460,59463,59464,59465,59471,59472,59476,59483,
+
59508,59509,59516,59517,59528,59529,59530,59534,59538,59547,59548,59549,59559,60443,
+
61005,61353,61838,61839,61968,62231,62235,62415,62606,63174,63175,63211,63231,63486,
+
63861,63924,64578,64692,64761,64783,65056,65099,65327,65328,65490,65640,65771,65916,
+                         66730,67679,72820,72939,79064,80698,86126),]
          ip device app channel os is_attributed Nov6 Nov7 Nov8 Nov9
5925    5314      1   2     477  6             0    0    0    3    1
5926    5348      1   2     477  6             0    1    2    2    1
5935   26995      1   2     477  6             0    0    0    2    1
5973   84896      1   2     477  6             0    0    1    2    0
9672   73516      1   3     153  9             0    0    0    3    0
9882  114904      2   2     205  9             0    0    1    3    0
12293 105292      1   2     205 10             0    0    0    2    1
16787   5348      1  18     107 13             0    0    1    2    1
16992  53454      1  18     107 13             0    0    1    2    0
17177 100275      1  18     107 13             0    0    1    2    1
17609   5314      1   3     115 13             0    0    2    2    0
17883   5348      1  18     121 13             0    0    1    2    2
17941  26995      1  18     121 13             0    0    1    2    1
18113 105560      1  18     121 13             0    0    0    3    0
23125  14615      1   2     205 13             0    0    0    2    1
23143  36150      1   2     205 13             0    0    1    4    0
23145  36213      1   2     205 13             0    0    2    2    0
23171  75595      1   2     205 13             0    0    2    3    1
23188 105519      1   2     205 13             0    0    1    3    1
23189 105534      1   2     205 13             0    0    1    2    0
23197 112061      1   2     205 13             0    0    0    3    0
23202 114904      1   2     205 13             0    0    1    4    0
23215 178851      1   2     205 13             0    0    1    2    0
23262 209663      2   2     205 13             0    0    1    2    0
24513  73487      1   9     234 13             0    0    2    2    1
24769  48212      1   2     237 13             0    0    2    2    2
25328  73516      1  12     245 13             0    0    1    2    0
25336  79857      1  12     245 13             0    0    0    2    1
25525   5314      1  15     245 13             0    0    1    3    0
25794  73516      1  15     245 13             0    0    2    2    0
25877  95766      1  15     245 13             0    0    1    2    0
26376  73487      1  12     259 13             0    0    1    2    1
```

26377	73516	1	12	259	13	0	0	2	2	2
27050	53454	1	12	265	13	0	0	0	3	0
27235	114276	1	12	265	13	0	0	0	3	0
27518	5314	1	26	266	13	0	0	0	3	0
27781	5348	1	3	280	13	0	0	1	2	1
27978	27001	1	3	280	13	0	0	1	2	0
28201	48170	1	3	280	13	0	0	0	2	1
28641	93587	1	3	280	13	0	0	0	3	0
28758	105560	1	3	280	13	0	0	2	2	0
28801	109620	1	3	280	13	0	0	1	2	0
28836	114276	1	3	280	13	0	0	1	4	1
28876	119531	1	3	280	13	0	0	0	2	1
28976	144739	1	3	280	13	0	0	0	2	1
29099	191851	1	3	280	13	0	0	1	2	0
30486	318333	1	2	364	13	0	0	0	3	0
34211	5314	1	9	466	13	0	0	1	3	1
34589	73487	1	2	469	13	0	0	1	2	0
34817	5425	1	2	477	13	0	0	1	2	0
43721	209663	1	2	205	17	0	0	2	2	0
47514	5314	1	18	121	18	0	0	0	2	1
49777	5348	1	3	280	18	0	0	1	2	2
51784	14737	1	15	3	19	0	0	0	4	0
51862	201182	1	15	3	19	0	1	1	2	0
52765	44067	1	18	107	19	0	0	1	2	0
52798	49602	1	18	107	19	0	0	1	3	1
52815	53454	1	18	107	19	0	0	2	2	1
55170	73516	1	21	128	19	0	0	1	2	1
57941	73516	1	3	153	19	0	0	1	6	3
58205	73516	1	23	153	19	0	1	1	2	0
58745	5314	1	12	178	19	0	0	1	2	0
59460	25648	1	2	205	19	0	0	1	2	0
59463	25737	1	2	205	19	0	1	1	3	2
59464	25761	1	2	205	19	0	0	2	3	0
59465	25792	1	2	205	19	0	1	1	2	0
59471	36150	1	2	205	19	0	0	1	5	1
59472	36183	1	2	205	19	0	0	0	2	1
59476	39756	1	2	205	19	0	0	2	5	1
59483	44645	1	2	205	19	0	0	1	2	0
59508	75595	1	2	205	19	0	0	2	2	1
59509	75634	1	2	205	19	0	0	1	2	1
59516	91694	1	2	205	19	0	0	2	1	2
59517	91712	1	2	205	19	0	0	1	3	0
59528	105292	1	2	205	19	0	0	0	2	1
59529	105323	1	2	205	19	0	0	2	2	2
59530	105433	1	2	205	19	0	2	1	2	1
59534	105534	1	2	205	19	0	0	2	1	2
59538	108341	1	2	205	19	0	0	1	3	2
59547	114490	1	2	205	19	0	1	2	2	0
59548	114878	1	2	205	19	0	0	2	4	3
59549	114904	1	2	205	19	0	1	2	2	1
59559	131635	1	2	205	19	0	0	2	2	0
60443	45745	1	2	219	19	0	0	1	2	0
61005	73487	1	9	234	19	0	0	0	5	3
61353	48282	1	2	237	19	0	0	2	3	1
61838	5314	1	12	245	19	0	0	1	2	0
61839	5348	1	12	245	19	0	0	0	2	1
61968	65687	1	12	245	19	0	0	1	2	0
62231	5314	1	15	245	19	0	0	2	3	0
62235	5348	1	15	245	19	0	0	2	2	0
62415	43793	1	15	245	19	0	0	1	2	0
62606	86767	1	15	245	19	0	1	1	2	0
63174	73487	1	12	259	19	0	0	0	5	2
63175	73516	1	12	259	19	0	0	0	2	1
63211	92673	1	12	259	19	0	0	1	2	0
63231	100182	1	12	259	19	0	0	0	3	0
63486	5348	1	25	259	19	0	0	1	2	0
63861	73487	1	12	265	19	0	2	2	2	2
63924	95766	1	12	265	19	0	0	1	4	1

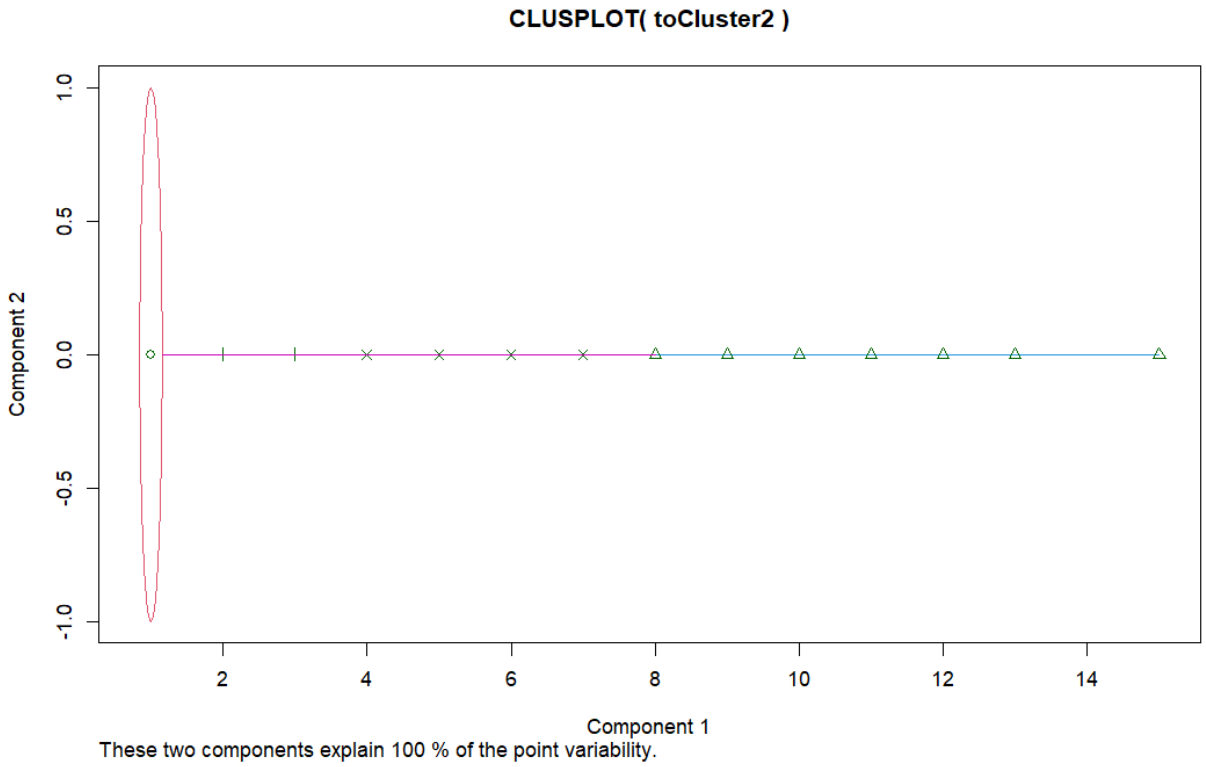
```
64578   5314      1   3     280 19            0    0    2    3    0
64692  17149      1   3     280 19            0    0    0    2    2
64761  25071      1   3     280 19            0    0    0    2    1
64783  26995      1   3     280 19            0    0    1    2    1
65056  49602      1   3     280 19            0    0    1    2    0
65099  53454      1   3     280 19            0    0    1    2    2
65327  73487      1   3     280 19            0    0    1    3    4
65328  73516      1   3     280 19            0    0    2    4    2
65490  88281      1   3     280 19            0    0    0    3    0
65640 102264      1   3     280 19            0    0    1    2    0
65771 114276      1   3     280 19            0    0    2    3    0
65916 137052      1   3     280 19            0    0    2    3    0
66730  73487      1  12     326 19            0    0    1    8    2
67679  55840      1   2     364 19            0    0    2    2    0
72820  73487      1   2     477 19            0    0    0    3    1
72939 114314      1   2     477 19            0    0    0    3    0
79064  73516      1  12     259 22            0    0    1    2    0
80698  59125      1   2     477 22            0    0    0    2    1
86126  73487      1   3     153 28            0    0    0    2    1
```

Using Overall Frequency

R^2 plotted for k from 1 to 14



Cluster model




```
> print(clu4)
K-means clustering with 4 clusters of sizes 96213, 18, 1352, 110

Cluster means:
      freq
1 1.000000
2 9.888889
3 2.118343
4 4.709091

Clustering vector:
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    4    1    1
  18   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
   1    1    1    1    1    1    1    1    1    1    1    1    1    3    1    1    1
  35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
  52   53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
  69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    3    1
  86   87   88   89   90   91   92   93   94   95   96   97   98   99  100  101  102
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153
```

```
  1   3   1   3   1   1   1   1   1   1   1   1   1   1   1   1   1
154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   3   1
171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204
  1   1   1   3   1   1   1   4   1   1   1   1   1   1   1   1   3
205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238
  1   1   1   1   1   1   1   1   1   1   1   1   1   3   1   1   1
239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
  1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1
256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
273 274 275 276 277 278 279 280 281 282 283 284 286 287 288 289 290
  1   3   1   1   1   1   1   1   3   4   1   1   1   1   1   1   1
291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358
  1   3   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375
  1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1
376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426
  1   3   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
478 479 480 481 483 484 485 486 487 488 489 490 491 492 493 494 495
  1   1   1   1   3   1   3   1   1   1   1   1   1   1   1   3   1
496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   3   1
513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529
  1   1   1   1   1   3   1   1   1   1   1   1   1   1   1   1   1
530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   3
547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580
  1   1   1   1   1   1   1   1   3   1   1   1   1   1   1   1   1
581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597
  1   1   3   1   1   3   1   1   1   1   1   1   1   1   1   1   1
598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614
  1   1   1   1   1   1   1   1   3   1   1   1   1   1   1   1   1
615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   3
666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682
  1   3   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716
  1   1   1   1   3   1   1   1   1   1   1   1   1   1   1   1   1
717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733
```

```
   1    1    1    1    1    1    1    4    1    1    1    1    1    1    1    1    1
 734  735  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 751  752  753  754  755  756  757  758  759  760  761  762  763  764  765  766  767
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 768  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800  801
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 802  803  804  805  806  807  808  809  810  811  812  813  814  815  816  817  818
   1    1    1    1    1    1    4    1    1    1    1    1    1    1    1    1    1
 819  820  821  822  823  824  825  826  827  828  829  830  831  832  833  834  835
   1    1    1    1    1    1    1    1    1    1    1    1    1    3    1    1    1
 836  837  838  839  840  841  842  843  844  845  846  847  848  849  850  851  852
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 853  854  855  856  857  858  859  860  861  862  863  864  865  866  867  868  869
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 870  871  872  873  874  875  876  877  878  879  880  881  882  883  884  885  886
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 887  888  889  890  891  892  893  894  895  896  897  898  899  900  901  902  903
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 938  939  940  941  942  943  944  945  946  947  948  949  950  951  952  953  954
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 955  956  957  958  959  960  961  962  963  964  965  966  967  968  969  970  971
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 989  990  991  992  993  994  995  996  997  998  999 1000 1001 1002
   1    1    1    1    1    1    1    1    1    1    1    1    1
 [ reached getOption("max.print") -- omitted 96693 entries ]

Within cluster sum of squares by cluster:
[1]   0.00000  71.77778 141.06509  86.69091
 (between_SS / total_SS =  93.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

IP addresses in cluster 2 and cluster 4

```
> # cluster 2
> consolidatedData[c(249,371,1293,3039,3582,4887,5507,5699,6054,7308,7543,8644,11703,
+                    12136,19497,22605,36942,37435),]
          ip device app channel os is_attributed freq
249   175837      1   2     205 19             0   12
371   114878      1   2     205 19             0    9
1293   73516      1  12     326 13             0   10
3039  209663      1   2     205 19             0    9
3582   73487      1  12     326 19             0   11
4887    5348      1   3     280 19             0   15
5507   73487      1   3     153 13             0   11
5699   73516      1  12     326 19             0   12
6054    5314      1   3     280 13             0    9
7308  108341      1   2     205 13             0    9
7543   39756      1   2     205 19             0    8
8644   73487      1   3     153 19             0   13
11703  73487      1   9     234 19             0    8
12136  73516      1   3     153 19             0   10
19497   5348      1  15     245 13             0    8
22605  73487      1   3     280 19             0    8
```

```
36942  73487      1  12     265 19             0    8
37435  73516      1   3     280 19             0    8

> # cluster 4
> consolidatedData[c(15,195,282,724,808,1378,1395,1899,1910,1927,2330,2373,2411,2614,
+
2635,2695,2837,2873,3007,3750,3906,3971,4032,4211,4403,4932,5624,5719,
+
5787,5823,5918,5984,6669,6911,7054,7151,7155,7213,7433,7879,7883,8116,
+
8697,8845,8948,9038,9195,9239,9442,10113,10345,10555,10814,10881,11146,11624,
+
11948,11969,12221,12429,12947,13021,13567,13935,14295,14887,15228,15503,15663,16471,
+
17226,17291,17678,17906,18311,18467,19264,19914,20834,21206,22552,23679,24033,24150,
+
25021,25147,25859,26236,28611,29036,29077,29662,30824,31017,31359,31830,33763,33946,
+
39608,40685,41847,42909,45908,46927,48019,55505,58390,59856,60595,66998),]
          ip device app channel os is_attributed freq
15     36150      1   2     205 13             0    5
195    26995      1   3     280 13             0    5
282     5314      1  18     107 19             0    6
724    25737      1   2     205 19             0    7
808    48212      1   2     237 13             0    6
1378  131635      1   2     205 19             0    4
1395   25705      1   2     205 19             0    5
1899    5314      1  21     232 19             0    4
1910   73516      1   3     153 13             0    5
1927  105560      1  15     245 19             0    4
2330    5348      1  12     178 13             0    4
2373   53454      1   3     280 19             0    5
2411  108341      1   2     205 19             0    6
2614   73487      1   9     234 13             0    5
2635  114276      1  15     245 19             0    4
2695   55690      1   2     364 19             0    4
2837    5348      1  18     107 19             0    5
2873    5314      1  15     245 13             0    4
3007   95766      1  12     265 19             0    6
3750   25792      1   2     205 19             0    4
3906    5348      1  21     128 19             0    4
3971  175837      1   2     205 13             0    7
4032   36150      1   2     205 19             0    7
4211  105560      1   3     280 13             0    4
4403  105323      1   2     205 19             0    6
4932   59125      1   2     205 13             0    6
5624   73487      1  12     259 13             0    4
5719   73516      1  23     153 19             0    4
5787    5314      1   3     280 19             0    5
5823    5314      0 151     347  0             0    6
5918    5348      1  12     328 19             0    4
5984   73516      1   9     234 19             0    4
6669   91694      1   2     205 19             0    5
6911   59125      1   2     205 19             0    5
7054   36213      1   2     205 13             0    4
7151   73516      1  15     245 13             0    4
7155  105433      1   2     205 13             0    4
7213  178873      1   2     205 19             0    7
7433    5348      1   2     477  6             0    6
7879   26995      1   3     280 19             0    4
7883    5314      1   3     280 18             0    4
8116    5314      1  21     128 19             0    7
8697    5314      1  15     245 19             0    5
8845   48240      1   3      30 19             0    4
8948    5314      1   2     477  6             0    4
9038   75595      1   2     205 19             0    5
9195  105456      1   2     205 13             0    4
9239   55910      1   2     364 19             0    4
```

9442	5348	1	18	121	13	0	5
10113	48282	1	2	237	19	0	6
10345	114904	2	2	205	9	0	4
10555	5314	1	9	466	13	0	5
10814	114904	1	2	205	19	0	6
10881	105519	1	2	205	13	0	5
11146	53454	1	18	107	19	0	5
11624	16741	1	3	280	13	0	5
11948	97773	1	15	245	13	0	4
11969	25761	1	2	205	19	0	5
12221	114904	1	2	205	13	0	5
12429	75634	1	2	205	13	0	5
12947	73516	1	12	259	13	0	6
13021	116713	1	2	364	19	0	4
13567	5348	1	3	280	13	0	4
13935	5314	1	3	115	13	0	4
14295	5314	1	8	145	19	0	4
14887	25679	1	2	205	13	0	4
15228	17149	1	3	280	19	0	4
15503	209663	1	2	205	17	0	4
15663	137052	1	3	280	19	0	5
16471	73487	1	15	245	19	0	5
17226	5348	1	18	107	13	0	4
17291	114490	1	2	205	19	0	5
17678	114276	1	3	280	19	0	5
17906	79881	1	3	280	19	0	4
18311	73487	1	12	259	19	0	7
18467	73487	1	12	178	13	0	5
19264	73487	1	12	259	18	0	4
19914	75634	1	2	205	19	0	4
20834	114276	1	3	280	13	0	6
21206	105456	1	2	205	19	0	5
22552	114276	1	9	244	19	0	4
23679	5314	1	13	477	18	0	4
24033	77655	1	3	280	19	0	4
24150	53454	1	9	134	13	0	5
25021	100275	1	18	107	13	0	4
25147	201182	1	15	3	19	0	4
25859	44744	1	3	280	19	0	4
26236	5314	1	23	153	19	0	5
28611	5348	1	3	280	18	0	5
29036	49602	1	18	107	19	0	5
29077	75595	1	2	205	13	0	6
29662	73487	1	2	477	19	0	4
30824	48282	1	7	101	19	0	4
31017	5314	1	12	265	13	0	4
31359	209663	1	2	205	13	0	4
31830	73516	1	21	128	19	0	4
33763	105534	1	2	205	19	0	5
33946	86767	1	15	245	19	0	4
39608	26995	1	18	121	13	0	4
40685	40216	1	3	280	19	0	4
41847	40631	1	3	280	19	0	5
42909	105433	1	2	205	19	0	6
45908	91574	1	2	205	19	0	4
46927	55840	1	2	364	19	0	4
48019	137052	1	3	280	13	0	4
55505	91712	1	2	205	19	0	4
58390	14737	1	15	3	19	0	4
59856	5348	1	12	265	19	0	4
60595	95766	1	18	121	13	0	4
66998	5348	1	15	245	19	0	4

Flagged IP addresses

5314	25761	44645	55910	88281	105323	114878	209663
5348	25792	44744	59125	91574	105433	114904	318333
5425	26995	45745	65687	91694	105456	116713	734871
14615	27001	48170	73487	91712	105519	119531	734873
14737	36150	48212	73516	92673	105534	131635	735160
16741	36183	48240	75595	93587	105560	137052	735162
17149	36213	48282	75634	95766	108341	144739	1758372
25071	39756	49602	77655	97773	109620	175837	
25648	40216	53454	79857	100182	112061	178851	
25679	40631	53485	79881	100275	114276	178873	
25705	43793	55690	84896	102264	114314	191851	
25737	44067	55840	86767	105292	114490	201182	