MS&E 226 - Small Data

Mini Project Report - Part III

SUNet ID: [dhruvj, tromero1]
Name: [Dhruv Joshi, Tyler Romero]

1 Testing Error and Analysis

As a brief review, Our dataset contains data sourced from Outbrain Inc, an online advertising services company, about the interaction of users with webpages on which ads were displayed by them. We have access to anonymized information about the webpage, the users and the advertisement topics. Our chosen target variables for prediction through a model are the time a user spends on page (continuous response variable or CRV) and whether or not the user clicks on an advertisement (Binary response variable or BRV).

1.1 CRV Model

Results Our proposed model for this system was a 2-step one. The first step was a binary classifier predicting whether the time on page was 0 or not. The second part takes the data that are predicted as having nonzero time on page and runs a linear regression model on them to predict the time spent on page in milliseconds. The reason behind choosing this two-step model was because more than half of the values for time on page were observed to be 0 - this is a case of the user 'bouncing' from the website. The actual origin of these zero value cases is unknown to us.

Part 1 The previously selected model was the full covariates model (.) and the threshold selected by ROC was $\lambda = 0.41$. For the best selected binary classifier model, the test set gave a 0-1 error of 0.2756985 (which is quite close to, but higher than, the cross-validation error on the same model which was 0.2690789. As a reference, the training error had a value of 0.2577354). We also obtained a false negative rate (FNR) of 0.35 - a rather large value. This 35% of users would essentially be considered to not contribute to potential ad revenue, and might lead to lower quality decision making by Outbrain management, should this model be used by them.

Part 2 The previously selected linear regression model gave an error of 200022183 ms, which is the same order of magnitude as, but 17% greater than, the training error observed (170766770 ms). As we had mentioned in our Part 2 analysis, this is very much expected, since using categorical variables to predict a continuous variable is a tricky problem to solve using linear regression. Our residuals plot in the previous report had also indicated that this is not a good linear classifier.

Analysis The testing error of the Part 1 model is similar to our cross-validation error, indicating this is a robust approach to estimating the generalization error of our models. The model itself, however, may not be the best choice and would need more work or perhaps a different methodology (discussed in more detail in the 'Summary' section of this report). The high FNR of 0.35 indicates that most of our users would be incorrectly predicted to bounce from the website, and from a business perspective, it would result in lost ad revenue for Outbrain. The performance of the linear classifier is more forgivable (from the perspective of Outbrain), but gives a large error when predicting the time that a user will spend on page. The residuals plot (Appendix 4.1.1) looks very similar to the one we had gotten previously.

1.2 BRV Model

Results When measured against our testing data, our logistic regression model (with covariates selected by forward stepwise selection and a prediction threshold of 0.27) achieved a zero-one error of 0.1924 and a precision of 0.3438.

Analysis In our previous report, we estimated our zero-one error to be 0.19, and our precision to be 0.33. Recall that we consider precision to be a better measure of our model's success than zero-one error, due to the fact that a zero-one error of 0.19 can be achieved by always predicting that an advertisement will not be clicked on. Our prediction of testing precision was very accurate (We were off by -4.18%). Observe that our actual testing precision was actually better than our predicted testing precision. We believe that this is due to chance: we just had a favorable test set.

2 Inference

Discussion of Coefficients After training a logistic regression model, insights can be gleaned from the statistical significance of that model's coefficients. See Appendix 4.1.1 for a table of coefficients for our logistic regression model. We say that a coefficient is statistically significant if the probability of observing that coefficient given that the true coefficient is zero (i.e. Null Hypothesis: $\beta = 0$, Alternative Hypothesis: $\beta \neq 0$) is less than α. Since this is a two-tailed test, that corresponds to witnessing a p-value less than $\alpha/2$. Using $\alpha = 0.05$, we determined that the coefficients corresponding to the following covariates were statistically significant (i.e. in these cases, we reject the null hypothesis that $\beta = 0$): *source_id, publisher_id, ad_id, category_confidence_level, platformmobile:publisher_id, display_id:ad_id, platformmobile:category_confidence_level, platformmobile:source_id*. We believe these results, it makes sense that the source, publisher, advertisement, etc. are correlated with whether or not a user clicks on an advertisement.

Coefficients on Testing Data We also trained our model on our test data, in order to see if our covariates still have a statistically significant effect. See Appendix 4.1.2 for these coefficient values and p-values. On our testing data, we found that only *source_id* was statistically significant.

A Model with All Available Covariates? Finally, we trained a new model using all available covariates on the training data, and found that the only covariate that was statistically significant was *platformmobile*. See Appendix 4.1.3 for details. This is clearly different from our model. We believe that this could be due to correlations between covariates that reduce the values of the coefficients, causing us to find less covariates to be non-zero at the $\alpha = 0.05$ level. Since our model was selected using forward stepwise selection, covariates that are highly correlated would be unlikely to appear together.

Bootstrap Estimation of Confidence Intervals The table below contains the bootstrap 95% confidence interval of each covariate. The bootstrap technique was used to calculate 1000 samples, from which the bias and standard error were calculated. ccl is used as an abbreviation for category_confidence_level. The values of zero in the Lower and Upper columns of the table may seem unusual at first glance. They are due to the fact that the function we used to calculate the confidence intervals rounds to 4 decimal places. The bias and standard error in these cases were simply too small. In every case, our bootstrap confidence interval contained the original value of the coefficient we calculated.

Covariate	Original	Bias	SE	Lower	Upper
Intercept	-7.622170e-01	-2.120852e-02	1.394770e-01	-1.0144	-0.4676
platformmobile	-2.536067e-01	2.224019e-02	9.715717e-02	-0.4663	-0.0854
platformtablet	-1.445233e-01	1.009248e-01	1.090642e-01	-0.4592	-0.0317
source_id	-3.446865e-05	3.775694e-06	6.086073e-06	-0.0001	0.0000
publisher_id	-6.352880e-04	-2.554331e-05	1.759161e-04	-0.0010	-0.0003
display_id	5.591029e-09	-4.776383e-10	1.127700e-08	0	0
ad_id	-1.966630e-06	-1.095087e-07	4.849363e-07	0	0
category_confidence_level	-3.964769e-01	4.684871e-02	1.357820e-01	-0.7095	-0.1772
traffic_sourcesearch	-9.978289e-02	-6.324546e-03	4.965728e-02	-0.1908	0.0039
traffic_sourcesocial	4.472789e-03	1.080808e-03	7.310744e-02	-0.1399	0.1467
platformmobile:publisher_id	2.677743e-04	6.745047e-05	6.191883e-05	0.0001	0.0003
platformtablet:publisher_id	8.119810e-05	4.704995e-05	1.049313e-04	-0.0002	0.0002
display_id:ad_id	8.174166e-14	-5.817004e-15	4.549258e-14	0	0
platformmobile:ccl	4.654978e-01	-8.497917e-02	8.775418e-02	0.3785	0.7225
platformtablet:ccl	3.615570e-02	-1.632059e-01	1.760220e-01	-0.1456	0.5444
ad_id:ccl	1.138878e-06	1.581166e-07	4.523245e-07	0	0
platformmobile:source_id	2.325912e-05	-2.415986e-06	7.073267e-06	0	0
platformtablet:source_id	1.902981e-05	2.866010e-07	1.076375e-05	0	0
publisher_id:ad_id	7.367445e-10	-5.379186e-11	5.185873e-10	0	0
publisher_id:ccl	2.833377e-04	-1.338150e-06	1.429222e-04	0.0000	0.0006

Potential Problems with Our Analysis Several of our covariates are correlated. For example, we performed a chi-squared test with null hypothesis that platform and traffic_source are not correlated, and obtained a p-value of 0.0004998 (much lower than our α of 0.05). This led us to reject our null-hypothesis in favor of our alternate hypothesis which was that platform and traffic_source are correlated. In addition, we performed a chi-squared test with regards to traffic_source and dayOfWeek, and obtained a p-value of 0.0104453. This led us to reject our null hypothesis (traffic_source and dayOfWeek are not correlated) in favor of our alternative hypothesis (traffic_source and dayOfWeek are correlated). Due to these correlated covariates, our inferences above may be confounded.

In general, we are more likely to incorrectly reject a null hypothesis when hypothesis tests are evaluated as part of a larger set. In order to counteract this, we could apply a Bonferroni correction. In essence, if we wish for an α of 0.05 for all of our hypotheses, we would divide α by the number of hypotheses we are testing. Then, in the case of training data coefficients, we would only reject our null hypotheses (that the coefficients are zero) in the cases of *source_id, publisher_id, ad_id*, and *platformmobile:category_confidence_level.*

The specific covariates that we select for our model influence which covariates we find to be significant. For example, observe the huge difference in p-value of ad_id in our model vs. the model including all covariates. In our model, we find ad_id to be significant, but in a model including all covariates, we would fail to find it significant. This could be because there is a covariate that ad_id is correlated with, and, when both of them appear together, it reduces the significance of each compared to when they appear individually.

Causal Relationships We are not convinced that these covariates exhibit any meaningful causal relationship on a macro level. Intuitively, none of our covariates cause any of our other covariates. This is because our covariates are simply labels and indicators. If there were causal relationships between covariates, the most likely candidate for a confounding variable is the *geo_location*, which we excluded from our models. A person's location may influence decisions like which platform (desktop/mobile) they use, and which time of day they are most likely to visit websites, and therefore click on certain ads. We would be hesitant to interpret any other variable as having a causal relationship. Such an inference at a macro level would most likely lead to incorrect assumptions about the underlying population model.

3 Summary

We chose to work on the highly industrially relevant ”big data” problem of predicting behaviour of users on the internet towards display advertisements. There is massive value associated with being able to predict which advertisements are most valuable to users. Using data and statistical learning to predict which ads are the most likely to generate revenue is a very dynamic problem and we wished to apply the techniques taught in this course to tackle this challenge using the dataset for the 2016 Kaggle competition.

3.1 Challenges faced and Recommendations for Data Collection

Magnitude of data The problem was interesting and challenging for several reasons. Firstly, the magnitude of the data generated is absolutely gargantuan. The data we chose to perform our experiments on was merely a much smaller (randomly sampled) subset of the original dataset - only 10,000 rows, for ease of computation. The original dataset consisted of over 2 billion ”events” and multiple tables representing the data (the database was in third normal form). This is for a sample collection interval of only 2 weeks. Understanding and drawing insights from such massive quantities of data is a big challenge and one of our driving forces was to understand how to tackle such challenges. Had we had access to the infrastructure to deal with data at this scale, we may have been able to draw better insights and make better predictions.

De-identified data To protect the identity of users, data is in an anonymized format. What this means is that user details, website details and topic details are not explicitly listed but are identified only by an alphanumeric code. (We assume that this code acts as a primary key for other tables which may contain more intimate details, which may have been collected by Outbrain by other means, but we do not have access to this metadata). This also implies that most of our variables were categorical - we had no continuous variables a priori, except the ”epoch” timestamp - i.e. the time in milliseconds when the event was captured, with respect to midnight, 1st January 1970. Working with data in this format was completely different from what we were used to dealing with in lectures, but was a valuable learning experience for us. We would recommend anyone building models on these datasets to have access to this information when doing so.

Categorical variables Building upon the previous two points made, our covariates are all categorical variables, which would be converted by a statistical package like R into multiple variables for each discrete value that this variable can take. This would lead to a massive amount of computation, and was the primary reason we chose to leave out a few variables with a large domain of discrete values (such as userid). Many of these values were only observed in a few (< 10) rows in the dataset, calling to question their utility. They were omitted from our analysis.

Sparseness of data Since the vast majority of users only see a particular advertisement a few times, if at all, our data is extremely sparse for a particular user - website - advertisement combination. On the other hand, a few users visit certain websites extremely often, and this causes significantly skewed data. In the world of online advertising, a 2% clickthrough rate is the industry average, which means we would have a very low tolerance for false negatives. We also observed that the bounce rate is generally very high, leading to us having access to a very small percentage of "conversion" events. Gleaning insights from such few events is a huge challenge.

3.2 Use in an Industrial context - Regular updation of the model

The average lifetime of online advertisements would be of the order of a few (2-4) weeks. Given the dynamic nature of the internet, it would be essential to regularly update models, for advertisements to remain relevant and maximize revenue expectations. From a Frequentist perspective, this would involve running experiments (similar to A/B tests) to determine which ads are performing better on different versions of the page, and using the p-value of a test statistic to determine which one to keep. Following a Bayesian approach, one would keep evolving the population model (whose prior is based on historic data) based on new data that would continually be collected based on how people are interacting with an advertisement. The posterior distribution so generated would be used to decide whether to keep an advertisement or remove it, based on the expectation value of revenue it would generate.

3.3 Exclusion of certain variables in our analysis

The userid (uuid) was removed as this was causing regression training times to go up by large magnitudes, since it is a categorical variable taking on alphanumeric values almost equal to n, the number of rows (extremely few users were repeated in the dataset). Hence this would cause $O(n)$ new binary covariates to be formed, which is also detrimental to the utility of the algorithm. This was removed prior to training. Similarly, publish_time (when the host webpage was published) and geo_location were not used in the initial analysis. For the latter, we justified the exclusion by observing a p-value for correlation between `geo_location` and our outcome variable `clicked` to be very high (> 0.5) under the null hypothesis.

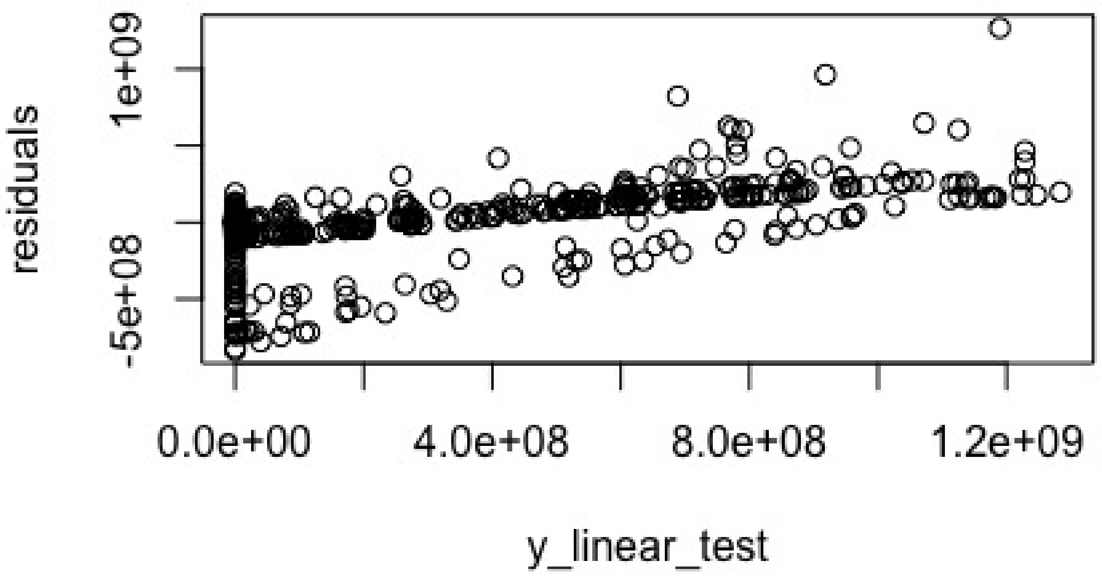
3.4 How we would re-attack the dataset

We would primarily explore the use of non-parametric models, which do not make any underlying assumptions about the population model, such as k-nearest neighbours, decision trees and random forests. KNN might be too computationally expensive (the curse of dimensionality), while decision trees can be prone to overfitting the training set. Random forest methods use randomness to choose covariates, thus preventing overfitting, and are the state of the art for sparse, high-dimensional datasets similar to ours.

4 Appendix

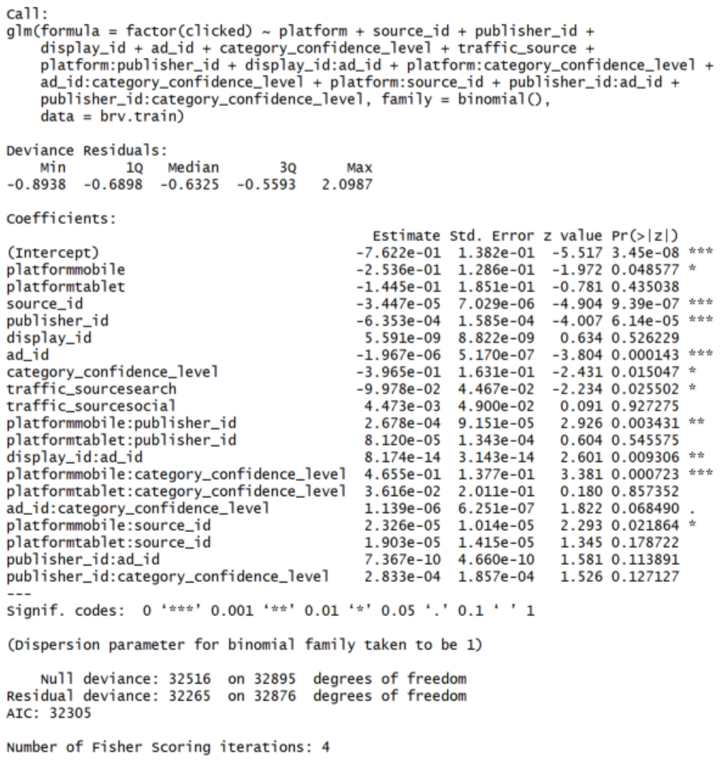
4.1 Selected model results on test data

4.1.1 Residuals plot for the continuous response model



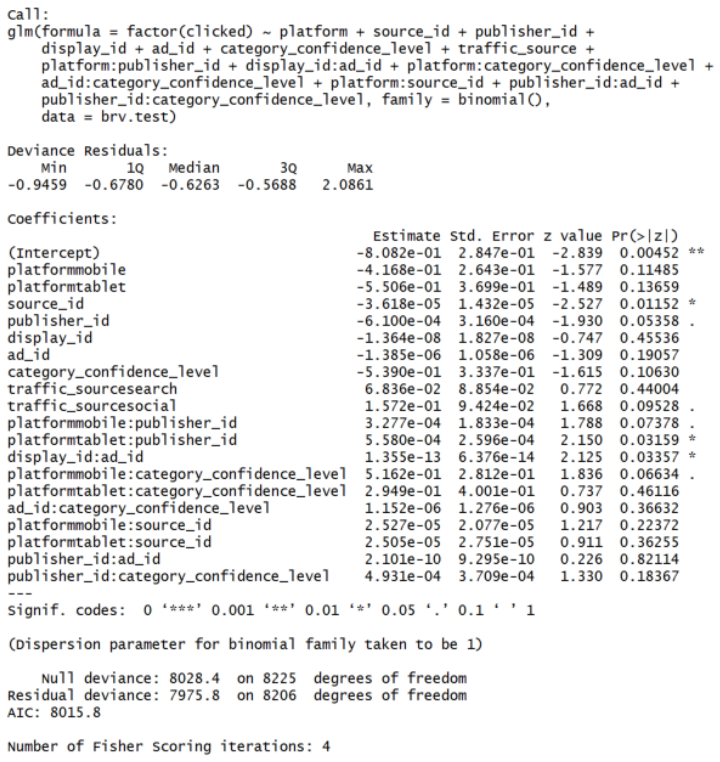
4.2 Logistic Model Coefficients

4.2.1 On Training Data w/ Chosen Subset of Covariates



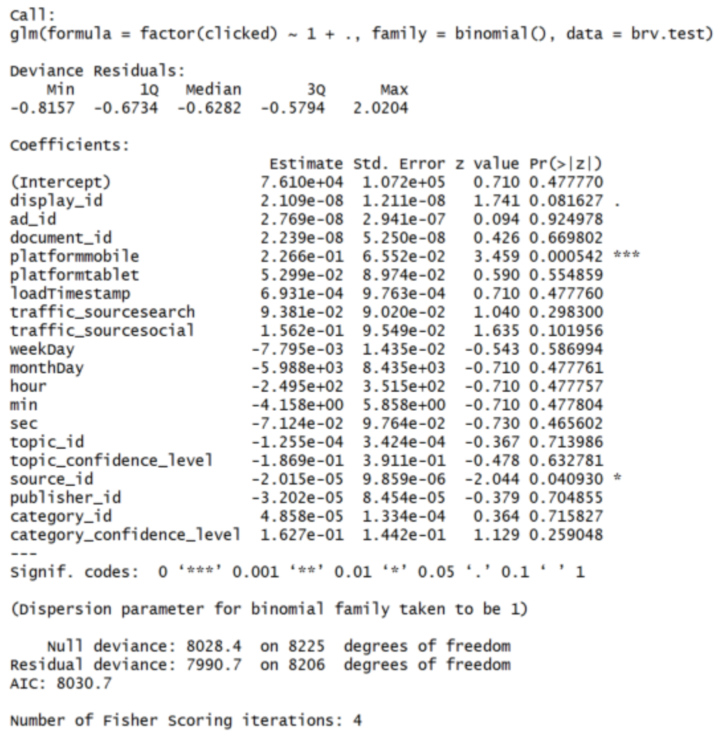
```
Call:
glm(formula = factor(clicked) ~ platform + source_id + publisher_id +
    display_id + ad_id + category_confidence_level + traffic_source +
    platform:publisher_id + display_id:ad_id + platform:category_confidence_level +
    ad_id:category_confidence_level + platform:source_id + publisher_id:ad_id +
    publisher_id:category_confidence_level, family = binomial(),
    data = brv.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.8938  -0.6898  -0.6325  -0.5593   2.0987

Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -7.622e-01  1.382e-01  -5.517 3.45e-08 ***
platformmobile                        -2.536e-01  1.286e-01  -1.972 0.048577 *
platformtablet                        -1.445e-01  1.851e-01  -0.781 0.435038
source_id                             -3.447e-05  7.029e-06  -4.904 9.39e-07 ***
publisher_id                          -6.353e-04  1.585e-04  -4.007 6.14e-05 ***
display_id                             5.591e-09  8.822e-09   0.634 0.526229
ad_id                                 -1.967e-06  5.170e-07  -3.804 0.000143 ***
category_confidence_level             -3.965e-01  1.631e-01  -2.431 0.015047 *
traffic_sourcesearch                  -9.978e-02  4.467e-02  -2.234 0.025502 *
traffic_sourcesocial                   4.473e-03  4.900e-02   0.091 0.927275
platformmobile:publisher_id            2.678e-04  9.151e-05   2.926 0.003431 **
platformtablet:publisher_id            8.120e-05  1.343e-04   0.604 0.545575
display_id:ad_id                       8.174e-14  3.143e-14   2.601 0.009306 **
platformmobile:category_confidence_level  4.655e-01  1.377e-01   3.381 0.000723 ***
platformtablet:category_confidence_level  3.616e-02  2.011e-01   0.180 0.857352
ad_id:category_confidence_level        1.139e-06  6.251e-07   1.822 0.068490 .
platformmobile:source_id               2.326e-05  1.014e-05   2.293 0.021864 *
platformtablet:source_id               1.903e-05  1.415e-05   1.345 0.178722
publisher_id:ad_id                     7.367e-10  4.660e-10   1.581 0.113891
publisher_id:category_confidence_level   2.833e-04  1.857e-04   1.526 0.127127
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 32516  on 32895  degrees of freedom
Residual deviance: 32265  on 32876  degrees of freedom
AIC: 32305

Number of Fisher Scoring iterations: 4
```

4.2.2 On Testing Data w/ Chosen Subset of Covariates



```
Call:
glm(formula = factor(clicked) ~ platform + source_id + publisher_id +
    display_id + ad_id + category_confidence_level + traffic_source +
    platform:publisher_id + display_id:ad_id + platform:category_confidence_level +
    ad_id:category_confidence_level + platform:source_id + publisher_id:ad_id +
    publisher_id:category_confidence_level, family = binomial(),
    data = brv.test)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9459  -0.6780  -0.6263  -0.5688   2.0861

Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -8.082e-01  2.847e-01  -2.839  0.00452 **
platformmobile                        -4.168e-01  2.643e-01  -1.577  0.11485
platformtablet                        -5.506e-01  3.699e-01  -1.489  0.13659
source_id                             -3.618e-05  1.432e-05  -2.527  0.01152 *
publisher_id                          -6.100e-04  3.160e-04  -1.930  0.05358 .
display_id                            -1.364e-08  1.827e-08  -0.747  0.45536
ad_id                                 -1.385e-06  1.058e-06  -1.309  0.19057
category_confidence_level             -5.390e-01  3.337e-01  -1.615  0.10630
traffic_sourcesearch                   6.836e-02  8.854e-02   0.772  0.44004
traffic_sourcesocial                   1.572e-01  9.424e-02   1.668  0.09528 .
platformmobile:publisher_id            3.277e-04  1.833e-04   1.788  0.07378 .
platformtablet:publisher_id            5.580e-04  2.596e-04   2.150  0.03159 *
display_id:ad_id                       1.355e-13  6.376e-14   2.125  0.03357 *
platformmobile:category_confidence_level  5.162e-01  2.812e-01   1.836  0.06634 .
platformtablet:category_confidence_level  2.949e-01  4.001e-01   0.737  0.46116
ad_id:category_confidence_level        1.152e-06  1.276e-06   0.903  0.36632
platformmobile:source_id               2.527e-05  2.077e-05   1.217  0.22372
platformtablet:source_id               2.505e-05  2.751e-05   0.911  0.36255
publisher_id:ad_id                     2.101e-10  9.295e-10   0.226  0.82114
publisher_id:category_confidence_level   4.931e-04  3.709e-04   1.330  0.18367
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8028.4  on 8225  degrees of freedom
Residual deviance: 7975.8  on 8206  degrees of freedom
AIC: 8015.8

Number of Fisher Scoring iterations: 4
```

4.2.3 On Training Data w/All Covariates



```
Call:
glm(formula = factor(clicked) ~ 1 + ., family = binomial(), data = brv.test)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.8157  -0.6734  -0.6282  -0.5794   2.0204

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)               7.610e+04  1.072e+05   0.710 0.477770
display_id                2.109e-08  1.211e-08   1.741 0.081627 .
ad_id                     2.769e-08  2.941e-07   0.094 0.924978
document_id               2.239e-08  5.250e-08   0.426 0.669802
platformmobile            2.266e-01  6.552e-02   3.459 0.000542 ***
platformtablet            5.299e-02  8.974e-02   0.590 0.554859
loadTimestamp             6.931e-04  9.763e-04   0.710 0.477760
traffic_sourcesearch      9.381e-02  9.020e-02   1.040 0.298300
traffic_sourcesocial      1.562e-01  9.549e-02   1.635 0.101956
weekDay                  -7.795e-03  1.435e-02  -0.543 0.586994
monthDay                 -5.988e+03  8.435e+03  -0.710 0.477761
hour                     -2.495e+02  3.515e+02  -0.710 0.477757
min                      -4.158e+00  5.858e+00  -0.710 0.477804
sec                      -7.124e-02  9.764e-02  -0.730 0.465602
topic_id                 -1.255e-04  3.424e-04  -0.367 0.713986
topic_confidence_level   -1.869e-01  3.911e-01  -0.478 0.632781
source_id                -2.015e-05  9.859e-06  -2.044 0.040930 *
publisher_id             -3.202e-05  8.454e-05  -0.379 0.704855
category_id               4.858e-05  1.334e-04   0.364 0.715827
category_confidence_level 1.627e-01  1.442e-01   1.129 0.259048
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8028.4  on 8225  degrees of freedom
Residual deviance: 7990.7  on 8206  degrees of freedom
AIC: 8030.7

Number of Fisher Scoring iterations: 4
```

5 Code Snippets in R

5.1 Code in R for Modeling BRV

```
library(data.table)
library(ggplot2)
library(dplyr)
library(MLmetrics)
library(cvTools)
library(glmnet)
# Testing the continuous response variable model on the test data
# FUNCTION DEFINITIONS
#
   --------------------------------------------------------------------------
# based on a threshold, decide which elements to call '0' and '1'
makePrediction = function(x, thresh = 0.5) {
  # @param x: This is the value/probability that your predictor outputs
  # @param thresh: the threshold that decides whether you classify as 1
     or 0
  # Outputs 1 or 0 depending on whether value is more than thresh or not
  # make a prediction on the probability of y, return 1 if greater than
     thresh, 0 otherwise
  if (x >= thresh)
    return(1)
  else
    return(0)
}

# Now will find the Confusion Matrix by using a fixed threshold
findCM = function(y_true, y, thresh) {
  # apply this function to the predicted values, and return a numeric
     vector
  y_observed <- as.numeric(lapply(y_pred, function(x)
     {makePrediction(x, thresh = thresh)}))

  # return confusion matrix
  cm <- ConfusionMatrix(y_observed, y_true)
  # print(cm)
  return(cm)
}
#
   --------------------------------------------------------------------------
# FIRST, read in and prepare the training data and re-train the best
   model
#
   --------------------------------------------------------------------------
## Part 1: All covariates, lambda = 0.41
Part1_bestModel <- notBounced ~ .
lambda = 0.41

crv.master <- fread("crv_train.csv")
crv.master <- select(crv.master, -V1) # remove the index column which
   automatically gets added

# preprocessing step: add new column which is a binarized version (1
   if nonzero) of time on page
crv.master <- mutate(crv.master, notBounced = ifelse(timeOnPage > 0,
   1, 0))

# the left out covariates take the longest time (-publish_time,-uuid,
   geo_location)
# timeOnPage is removed because it would be cheating
X_train <- select(crv.master, -uuid, -publish_time, -geo_location,
   -timeOnPage) # this is our actual p+1 covariates set

# training the model
binary_timeOnPage <- glm(Part1_bestModel, data = X_train)

# SECOND, reading in and converting the data into the format usable by
   the models
testData <- fread("crv_test.csv")
testData <- select(testData, -V1) # remove the index column which
   automatically gets added

# adding the notBounced column
testData <- mutate(testData, notBounced = ifelse(timeOnPage > 0, 1, 0))

# Finally, prepare the design matrix X
X_test <- select(testData, -uuid, -publish_time, -geo_location,
   -timeOnPage)

# Running the model on the test data
y_pred <- predict(binary_timeOnPage, X_test, type = "response") #
   predicted response
y_truth <- as.numeric(X_test$notBounced)
y_zeroOne <- mean(as.numeric(lapply(y_pred, function(x)
   {makePrediction(x, thresh = lambda)})))
cm <- findCM(y = y_pred, y_true = y_truth, thresh = lambda)
# get values of precision
tn = cm[1]
fn = cm[2]
fp = cm[3]
tp = cm[4]
print("Confusion Matrix:")
print(cm)
cat("FNR = ", fn/(tp + fn))

y_pred_train <- predict(binary_timeOnPage, X_train, type = "response")
   # these are probabilities
Y_train <- select(mutate(as.data.frame(y_pred_train), predicted =
   ifelse(y_pred_train > 0.5, 1, 0)), predicted)
training_error <- sum(abs(Y_train - X_train$notBounced))/nrow(X_train)
```

```
cat('training error:', training_error, '\n')

y_pred_test <- predict(binary_timeOnPage, X_test, type = "response")
Y_test <- select(mutate(as.data.frame(y_pred_test), predicted =
   ifelse(y_pred_test > 0.5, 1, 0)), predicted)
testing_error <- sum(abs(Y_test - X_test$notBounced))/nrow(X_test)
cat('testing error:', testing_error, '\n')


#
   ------------------------------------------------------------------------------
## Part 2:
#
   ------------------------------------------------------------------------------
# Our winning model...
Part2_bestModel <- timeOnPage ~ display_id + document_id +
   traffic_source +
category_confidence_level +
  platform + loadTimestamp + topic_id + topic_confidence_level +
  display_id:document_id + display_id:traffic_source +
  document_id:platform +
  display_id:platform + display_id:category_confidence_level +
  traffic_source:category_confidence_level + document_id:traffic_source
+
  document_id:loadTimestamp + document_id:category_confidence_level +
  platform:loadTimestamp + display_id:topic_id + document_id:topic_id +
  document_id:topic_confidence_level

# Generate the training set and re-train the model
timePage.train <- filter(crv.master, notBounced == 1)
timePage.train <- select(timePage.train, -notBounced)

X_linear_train <- select(timePage.train, -uuid, -publish_time,
   -geo_location)
y_linear_train <- X_linear_train$timeOnPage

# Prepare the test data
X_linear_test <- filter(testData, notBounced == 1)
y_linear_test <- X_linear_test$timeOnPage

# Train and make predictions!
lm_timeOnPage <- lm(Part2_bestModel, data = X_linear_train)

# Find training error and testing error for each of these models
# Convert to 0 and 1 based on threshold level 0.5 (as dataframe, then
   convert back into numeric vector)
y_pred_train <- predict(lm_timeOnPage, X_linear_train, type =
   "response") # these are probabilities
training_error <- sqrt(mean((y_pred_train - y_linear_train)^2))
cat('training error:', training_error, '\n')
```

```
y_pred_test <- predict(lm_timeOnPage, X_linear_test, type = "response")
testing_error <- sqrt(mean((y_pred_test - y_linear_test)^2))
cat('testing error:', testing_error, '\n')

# Plot residuals
residuals <- y_linear_test - y_pred_test
plot(y_linear_test, residuals)

#
   ----------------------------------------------------------------------------------
```

5.2 Code in R for BRV Model Testing

```
library(data.table)
library(ggplot2)
library(dplyr)
library(MLmetrics)
library(leaps)
library(glmnet)
library(boot)
library(MASS)

input_directory <- ".\\input"
setwd(input_directory)

#----------- Threshold Helper -----------------
predictThreshGlm <- function(prediction, K) {
  return(as.numeric(prediction > K))
}


#----------- Get the training and testing data ------------
brv <- fread("brv.csv")
brv <- select(brv, -c(V1))
brv$clicked <- as.factor(brv$clicked)

#split data
set.seed(851)
train.ind = sample(nrow(brv), 4*round(nrow(brv)/5)) #80% of data is
   for training, 20% for test
brv.train = brv[train.ind,]
brv.test = brv[-train.ind,]


#----------- Choose Covariates for Logistic Regression ------------
#Remove HUGE factor covariates
brv.train <- select(brv.train, -publish_time, -geo_location)
brv.test <- select(brv.test, -publish_time, -geo_location)

#Generate model using forward stepwise selection
```

```
min.model <- glm(factor(clicked) ~ 1, data = brv.train, family =
    binomial())
biggest <- formula(glm(factor(clicked) ~ .:., brv.train, family =
    binomial()))
fwd.model <- step(min.model, direction='forward', scope=biggest)


#----------- Get Testing Error and Precision -------------
glm.train <- fwd.model
rawPre <- predict(glm.train, brv.test, type="response")

threshold <- 0.27
pre1 <- predictThreshGlm(rawPre, K=threshold)

cm <- as.matrix(ConfusionMatrix(y_pred = pre1, y_true =
    brv.test$clicked))
if(!("0" %in% colnames(cm))){
  cm <- cbind(c(0,0),cm)
} else if(!("1" %in% colnames(cm))) {
  cm <- cbind(cm, c(0,0))
}
fp <- cm[3]
tp <- cm[4]

zeroOne <- ZeroOneLoss(pre1, brv.test$clicked)
precision = tp/(tp+fp)

print("Zero-One Testing Error:")
print(zeroOne)

print("Test Set Precision")
print(precision)

#Formula chosen by forward stepwise selection
fss <- factor(clicked) ~ platform + source_id + publisher_id +
  display_id + ad_id + category_confidence_level + traffic_source +
  platform:publisher_id + display_id:ad_id +
      platform:category_confidence_level +
  ad_id:category_confidence_level + platform:source_id +
      publisher_id:ad_id +
  publisher_id:category_confidence_level


glm.test <- glm(formula = fss, family = binomial(), data = brv.test)

glm.all <- glm(formula = factor(clicked) ~ 1 + ., family = binomial(),
    data = brv.test)

print("Coefficients of glm on training data")
print(summary(glm.train))
print("Coefficients of glm on testing data")
print(summary(glm.test))
print("All available coefficients on training data")
print(summary(glm.all))


#--------- Bootstrapping Estimation of Coefficient Confidence
   Intervals -------------

#Boostrap estimation of regression coefficient confidence intervals
#From http://www.statmethods.net/advstats/bootstrapping.html
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- glm(formula, family = binomial(), data = d)
  return(coef(fit))
}

results <- boot(data = brv.train, statistic = bs, R = 10, formula =
   fss)

print(fss)
print(results)

# get 95% confidence intervals
for(i in 1:20) {
  print(i)
  print(boot.ci(results, type="norm", index=i))
}

#---------- Correlation Tests -----------------------
tbl <- table(brv$platform, brv$traffic_source)
chi1 <- chisq.test(tbl, simulate.p.value = TRUE)


tbl <- table(brv$weekDay, brv$traffic_source)
chi2 <- chisq.test(tbl, simulate.p.value = TRUE)


tbl <- table(brv$geo_location, brv$clicked)
chi3 <- chisq.test(tbl)
```