# Introduction

The project involved working on the Avazu CTR competition that can be found [here](https://www.kaggle.com/c/avazu-ctr-prediction) on Kaggle.

The problem statement was to predict whether an ad would be clicked or not based on a set of categorical predictors provided for a set of ads that formed the training sample.

More details on the data can be found on the Kaggle competition page.

# Data Handling and Tools

The training data was 6 GB in size containing about 40M rows, i.e. 40M ad impressions. That was found to be too big to fit in the memory (and work upon, say, like a data frame) and conduct data analysis on. Hence I sampled the data to analyze and design features and algorithms.

The data was sampled into the following four buckets:

1. TrainSample: Randomly sampled ~2.2M rows over the first 8 days
2. OOSVal: Randomly sampled ~1M rows over the first 8 days
3. OOTVal: Randomly sampled ~2M rows on the 9th day
4. TestSample: Randomly sampled ~2M rows on the 10th day

The code for this sampling is attached here: 

The objective of this sampling was to have available a training dataset, TrainSample, on which exploratory analysis as well as any model building will be done. The OOSVal can be used to test the features and optimize on model parameters. The OOTVal could then be used to conduct a validation of the algorithm and also allow for optimized aggregation of all models, i.e. create an ensemble. And then finally, the TestSample would be used to test the model performance.  
Ideally I would have liked to conduct a thorough cross validation in the TrainSample itself however the size of the data and limited resources did not allow sufficient room to do so.

The tools used were: Ipython notebook server, Python and R. Most of the analysis was done in R. Python was used as a scripting tool to create the samples as stated above and to obtain summary statistics of the data.

To be noted that in the document below:variables represent the columns of the train/test data. Categories denote the unique values those variables can contain.  
CTR stands for Click Through Rate.

# Approach

With the objective as stated on the Kaggle competition and the time being 7 days my idea was to cast my net wide and try as many approaches to mine the data as possible. I attempted to explore different approaches without going into details or depth as the objective was stated as to understand the thought process of approaching such a problem and also come up with as many features.

There were no numeric variables on the data. Hence none of the variables had any order, except for “hour” that shows the day and the hour of each ad. With all the other variables being categories the the approach I took was to identify the most significant variables and the most significant interactions. Also to keep in mind is that the data could also end up being really sparse with large number of categories and hence could be prone to overfitting. With that in mind the following was the process I followed.

# Data Summary

## Training Data

For this portion I ran the script *CountDFScript.py* that was used to obtain the count and number of clicks for each value of each of the categorical variables. The output of this script is in *CountDFScript.html*. As can be seen, variables related to Apps and Site seem to have a wide range of CTR compared to others.



## Overlap between Training and Test

Since all the data is categorical the model would only work if there is a strong overlap between the training and the test set. I looked at that overlap as shown below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **NumTrainCat** | **NumTestCat** | **TestCatinTrain** | **TestinTrain** | **TrainCommonCount** |
| **1** | banner\_pos | 7 | 7 | 7 | 100.0% | 100.0% |
| **2** | C1 | 7 | 7 | 7 | 100.0% | 100.0% |
| **3** | C15 | 8 | 8 | 8 | 100.0% | 100.0% |
| **4** | C16 | 9 | 9 | 9 | 100.0% | 100.0% |
| **5** | C18 | 4 | 4 | 4 | 100.0% | 100.0% |
| **6** | DayOfWeek | 7 | 1 | 1 | 100.0% | 12.0% |
| **7** | device\_conn\_type | 4 | 4 | 4 | 100.0% | 100.0% |
| **8** | device\_type | 5 | 4 | 4 | 100.0% | 100.0% |
| **9** | Hour | 24 | 24 | 24 | 100.0% | 100.0% |
| **10** | C20 | 166 | 161 | 159 | 98.8% | 99.4% |
| **11** | app\_category | 29 | 26 | 25 | 96.2% | 100.0% |
| **12** | site\_category | 22 | 21 | 20 | 95.2% | 100.0% |
| **13** | device\_model | 5880 | 5133 | 4748 | 92.5% | 99.8% |
| **14** | C19 | 61 | 48 | 43 | 89.6% | 94.7% |
| **15** | site\_id | 3064 | 2489 | 2202 | 88.5% | 99.3% |
| **16** | C21 | 52 | 40 | 33 | 82.5% | 88.7% |
| **17** | site\_domain | 3656 | 2829 | 2329 | 82.3% | 99.6% |
| **18** | app\_domain | 267 | 154 | 125 | 81.2% | 99.9% |
| **19** | app\_id | 4162 | 2808 | 2163 | 77.0% | 98.6% |
| **20** | C17 | 365 | 221 | 167 | 75.6% | 70.4% |
| **21** | C14 | 2039 | 1158 | 801 | 69.2% | 65.3% |
| **22** | device\_id | 305577 | 176535 | 13221 | 7.5% | 83.8% |

*NumTrainCat* represents the number of categories for that variable in Training Sample

*NumTestCat:* the number of categories for that variable in Test Sample

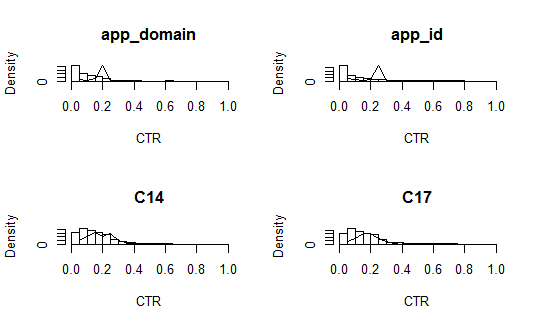
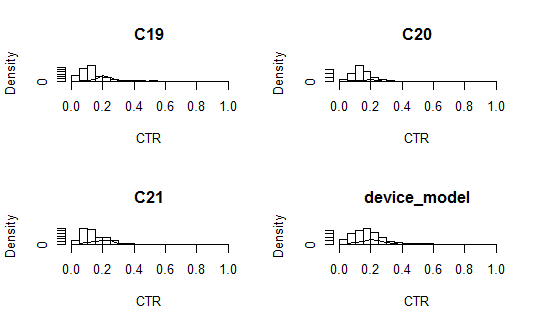
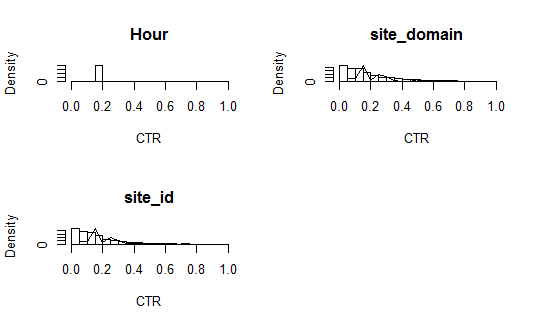
*TestCatinTrain:* number of test categories that are also in train.

*TestinTrain:* Ratio of Test categories that are also in Train

*TrainCommoncount*: The ratio of total Train observations that are covered by the categories common between Test and Train.

We see that the variables with less number of categories are (quite intuitively) 100% same in Test and Train while variables with many categories have less of an overlap. To note is device\_id that seems to have just a 7.5% overlap leading to believe that the data does not contain a consistent set of users through the days.

Next I went further and looked at the histogram of CTR for each category for each variable that has more than 20 categories and also looked at the number of samples by category for the same variable. I plotted that information together as seen below. This gave an idea that for each variable the number of samples peaks for those categories where CTR is about 20%. The CTR for the overall population was found to be 17%.



We can see above that C19 seems to peak at 20% instead of ~17% that most of the variables seem to peak at. When looking at the CTR for the top three populous categories for each variable we get the table below. Looking at the highlighted rows we can see that C14 and C19 have the highest CTR (>24%) for their most populous category.

|  | **Colname** | **VarValue** | **Clicks** | **Freq** | **CTR** |
| --- | --- | --- | --- | --- | --- |
| **1** | app\_category | 07d7df22 | 300862 | 1498997 | 0.20070887 |
| **2** | app\_category | 0f2161f8 | 57969 | 530556 | 0.10926085 |
| **3** | app\_category | cef3e649 | 10096 | 101705 | 0.09926749 |
| **4** | app\_domain | 7801e8d9 | 306166 | 1559265 | 0.19635277 |
| **5** | app\_domain | 2347f47a | 33997 | 255884 | 0.13286098 |
| **6** | app\_domain | 5c5a694b | 12289 | 61486 | 0.19986664 |
| **7** | app\_id | ecad2386 | 295962 | 1478532 | 0.20017287 |
| **8** | app\_id | e2fcccd2 | 12289 | 61475 | 0.19990240 |
| **9** | app\_id | 7358e05e | 7510 | 37793 | 0.19871405 |
| **10** | banner\_pos | 0 | 271968 | 1630703 | 0.16677960 |
| **11** | banner\_pos | 1 | 115174 | 631828 | 0.18228695 |
| **12** | banner\_pos | 7 | 776 | 2233 | 0.34751455 |
| **13** | C1 | 1005 | 355353 | 2079472 | 0.17088617 |
| **14** | C1 | 1002 | 26947 | 126415 | 0.21316299 |
| **15** | C1 | 1010 | 4742 | 51084 | 0.09282750 |
| **16** | C14 | 4687 | 14306 | 57378 | 0.24932901 |
| **17** | C14 | 20108 | 9344 | 38029 | 0.24570722 |
| **18** | C14 | 20093 | 7237 | 12511 | 0.57845096 |
| **19** | C15 | 320 | 336564 | 2111126 | 0.15942393 |
| **20** | C15 | 300 | 48711 | 130452 | 0.37340171 |
| **21** | C15 | 216 | 2472 | 20001 | 0.12359382 |
| **22** | C16 | 50 | 339226 | 2130588 | 0.15921708 |
| **23** | C16 | 250 | 44750 | 104811 | 0.42695900 |
| **24** | C16 | 36 | 2472 | 20001 | 0.12359382 |
| **25** | C17 | 1722 | 54186 | 282536 | 0.19178441 |
| **26** | C17 | 423 | 14306 | 57378 | 0.24932901 |
| **27** | C17 | 1994 | 14062 | 35481 | 0.39632479 |
| **28** | C18 | 0 | 152680 | 946759 | 0.16126596 |
| **29** | C18 | 2 | 116330 | 391796 | 0.29691472 |
| **30** | C18 | 3 | 113360 | 759140 | 0.14932687 |
| **31** | C19 | 39 | 127699 | 525185 | 0.24315051 |
| **32** | C19 | 35 | 118134 | 698795 | 0.16905387 |
| **33** | C19 | 167 | 29386 | 182014 | 0.16144912 |
| **34** | C20 | -1 | 204600 | 1054191 | 0.19408248 |
| **35** | C20 | 100084 | 31081 | 142371 | 0.21830991 |
| **36** | C20 | 100148 | 22482 | 96087 | 0.23397546 |
| **37** | C21 | 23 | 100986 | 473104 | 0.21345412 |
| **38** | C21 | 79 | 54800 | 287382 | 0.19068696 |
| **39** | C21 | 221 | 44641 | 263070 | 0.16969248 |
| **40** | DayOfWeek | Tue | 106410 | 658739 | 0.16153590 |
| **41** | DayOfWeek | Wed | 58837 | 374349 | 0.15717152 |
| **42** | DayOfWeek | Thu | 49416 | 271135 | 0.18225607 |
| **43** | device\_conn\_type | 0 | 358023 | 1961940 | 0.18248417 |
| **44** | device\_conn\_type | 2 | 24855 | 177005 | 0.14041976 |
| **45** | device\_conn\_type | 3 | 5193 | 124917 | 0.04157160 |
| **46** | device\_model | 8a4875bd | 19406 | 138098 | 0.14052340 |
| **47** | device\_model | d787e91b | 17946 | 77972 | 0.23015954 |
| **48** | device\_model | 1f0bc64f | 17945 | 78272 | 0.22926462 |
| **49** | device\_type | 1 | 356453 | 2088916 | 0.17064018 |
| **50** | device\_type | 0 | 26947 | 126415 | 0.21316299 |
| **51** | device\_type | 4 | 4005 | 43763 | 0.09151566 |
| **52** | Hour | 13 | 23098 | 136346 | 0.16940724 |
| **53** | Hour | 14 | 21708 | 120557 | 0.18006420 |
| **54** | Hour | 12 | 21439 | 123401 | 0.17373441 |
| **55** | site\_category | f028772b | 128358 | 716831 | 0.17906313 |
| **56** | site\_category | 50e219e0 | 114452 | 897061 | 0.12758553 |
| **57** | site\_category | 28905ebd | 90389 | 427701 | 0.21133689 |
| **58** | site\_domain | c4e18dd6 | 98653 | 817365 | 0.12069638 |
| **59** | site\_domain | f3845767 | 78851 | 377453 | 0.20890283 |
| **60** | site\_domain | 7e091613 | 46987 | 184044 | 0.25530308 |
| **61** | site\_id | 85f751fd | 92180 | 787885 | 0.11699677 |
| **62** | site\_id | 1fbe01fe | 78851 | 377453 | 0.20890283 |
| **63** | site\_id | e151e245 | 42728 | 144599 | 0.2954930 |

# Modeling

In the Data Summary section above I was able to find different cuts of the data using single variables that seemed to yield some predictive power. I also then considered looking at variables that have categories that, by themselves, have either a very low or very high CTR as seen below.

> aggregate(Freq~Colname, temp, sum)

Colname Freq

1 app\_id 1352

2 C14 123

3 C17 234

4 device\_model 13

5 site\_domain 117

6 site\_id 950

> temp = FreqDFTrain[FreqDFTrain$CTR<=0.02,]

> aggregate(Freq~Colname, temp, sum)

Colname Freq

1 app\_domain 2003

2 app\_id 142443

3 C14 60762

4 C17 101877

5 device\_model 71326

6 device\_type 2

7 site\_category 198

8 site\_domain 45682

9 site\_id 64006

We see that we can find variables with categories that have a low CTR or even a high CTR. But that would not help create the most optimized “cuts” in the data so as to have a prediction for the overall population. At this point I realized that classification algorithms could probably help better in extracting this information for each category and then using that to make a prediction in the val/test sample.

## Association Analysis

We know that we cannot find correlation between these variables as they are categorical. Hence I used the Chi-square test to determine the degree of association between each pair of variables. This can be found here:



In the association analysis we see that click has the highest degree of association with site\_id, site\_domain, C14 and C17. Also we see that both these pairs of variables are also strongly correlated to each other. We also see that Hour and DayofWeek are not strongly associated with any of the variables leading me to believe that there does not seem to be a strong predictive power when looking chronologically through the data.

## Feature Generation

I looked at two aspects for feature generation:

1. Feature generation over chronological data – I believed that since the data in this problem will be sourced as batches and can be implemented in a time sequential manner I wanted to explore information that can be predictive as a time series. For this purpose I looked at a sample case where I analyzed the time series of CTR grouped by app\_category, app\_domain and another where I grouped by site\_category.

|  | **app\_category** | **app\_domain** | **Sample Size** | **Corr** |
| --- | --- | --- | --- | --- |
| **1** | 07d7df22 | 7801e8d9 | 168 | -0.4341030370 |
| **2** | 0f2161f8 | 2347f47a | 168 | 0.0576275657 |
| **3** | 0f2161f8 | 33da2e74 | 168 | 0.1347206325 |
| **4** | 0f2161f8 | 5c5a694b | 168 | -0.1770657689 |
| **5** | 0f2161f8 | 7801e8d9 | 168 | 0.1907375039 |
| **6** | 0f2161f8 | 82e27996 | 168 | -0.0490509147 |
| **7** | 0f2161f8 | ae637522 | 168 | 0.0351961042 |
| **8** | 0f2161f8 | aefc06bd | 168 | 0.4148870465 |
| **9** | 0f2161f8 | d9b5648e | 168 | 0.1954818149 |
| **10** | 8ded1f7a | 2347f47a | 168 | -0.1683532881 |
| **11** | cef3e649 | 2347f47a | 168 | -0.0937035644 |
| **12** | cef3e649 | 7801e8d9 | 168 | -0.1670764081 |
| **13** | cef3e649 | b9528b13 | 168 | 0.4692178834 |
| **14** | f95efa07 | 2347f47a | 168 | -0.1350490596 |
| **15** | cef3e649 | ae637522 | 167 | 0.0205784381 |
| **16** | 0f2161f8 | 5b9c592b | 165 | -0.0339622653 |
| **17** | 8ded1f7a | b5f3b24a | 165 | 0.0577928006 |
| **18** | 0f2161f8 | 885c7f3f | 164 | 0.0957198320 |
| **19** | cef3e649 | d9b5648e | 163 | 0.0391135156 |
| **20** | d1327cf5 | 2347f47a | 163 | -0.0174904227 |

The last column, corr, provides the correlation of time series of CTR with time and we can see that there does not seem to be a very strong relationship through time. The same can be seen in the trend when grouped by site\_category also.

|  | **site\_category** | **Sample Size** | **Corr** |
| --- | --- | --- | --- |
| **1** | 28905ebd | 168 | -0.277452332 |
| **2** | 3e814130 | 168 | -0.265334750 |
| **3** | 50e219e0 | 168 | 0.339176998 |
| **4** | 75fa27f6 | 168 | -0.262117547 |
| **5** | f028772b | 168 | -0.219668469 |
| **6** | f66779e6 | 168 | -0.152497981 |
| **7** | 335d28a8 | 167 | -0.139642770 |
| **8** | c0dd3be3 | 162 | -0.033821658 |
| **9** | 76b2941d | 137 | 0.049206024 |
| **10** | 72722551 | 132 | -0.019107638 |
| **11** | dedf689d | 127 | -0.423382924 |
| **12** | 0569f928 | 122 | 0.006519084 |
| **13** | 70fb0e29 | 119 | -0.244917396 |
| **14** | a818d37a | 78 | 0.078454535 |
| **15** | 42a36e14 | 66 | 0.005823837 |
| **16** | bcf865d9 | 57 | 0.054269777 |
| **17** | e787de0e | 22 | -0.142757620 |
| **18** | 9ccfa2ea | 21 | -0.332347026 |

1. Feature Generation using summary statistics – Here I looked at the number of records and CTR grouped by each category for all variables. I analyzed if the number of records can have any relationship with the CTR. The correlation below represents the correlation between CTR and #Records corresponding to each category grouped by the variable. It can be seen that no correlation is high enough and the two that are, DayofWeek and device\_conn\_type, do not seem intuitive enough to move forward.

|  |  |
| --- | --- |
| Colname | Corr |
| app\_category | 26% |
| app\_domain | 2% |
| app\_id | 0% |
| banner\_pos | -11% |
| C1 | 42% |
| C14 | 3% |
| C15 | -17% |
| C16 | -16% |
| C17 | 6% |
| C18 | 22% |
| C19 | 13% |
| C20 | 5% |
| C21 | 22% |
| DayOfWeek | -73% |
| device\_conn\_type | 79% |
| device\_id | 0% |
| device\_model | -1% |
| device\_type | 42% |
| Hour | -8% |
| site\_category | 23% |
| site\_domain | 0% |
| site\_id | 0% |

1. Feature Generation for each user (device\_id) – I wanted to see if there are certain users who are much more prone to clicking than others. For that purpose I looked at the CTR broken by device\_id the results of which are shown below. It can be seen that the CTRs are quite low in value and hence device\_id may not add a lot of predictive power when used to split the data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Device\_id** | **Clicks** | **TotalRecords** | **CTR** | **IDinTest** |
| **1** | a99f214a | 327685 | 1871751 | 0.175069 | TRUE |
| **2** | c357dbff | 880 | 1377 | 0.63907 | FALSE |
| **3** | 936e92fb | 48 | 787 | 0.060991 | TRUE |
| **4** | afeffc18 | 120 | 475 | 0.252632 | TRUE |
| **5** | 0f7c61dc | 364 | 471 | 0.772824 | TRUE |
| **6** | b09da1c4 | 35 | 243 | 0.144033 | TRUE |
| **7** | d857ffbb | 60 | 238 | 0.252101 | TRUE |
| **8** | 987552d1 | 0 | 235 | 0 | TRUE |
| **9** | cef4c8cc | 53 | 221 | 0.239819 | TRUE |
| **10** | 28dc8687 | 0 | 197 | 0 | FALSE |
| **11** | 03559b29 | 0 | 125 | 0 | TRUE |
| **12** | d2e4c0ab | 0 | 108 | 0 | FALSE |
| **13** | 02da5312 | 16 | 103 | 0.15534 | TRUE |
| **14** | abab24a7 | 0 | 88 | 0 | FALSE |
| **15** | f1d9c744 | 0 | 88 | 0 | FALSE |
| **16** | eec6d022 | 0 | 84 | 0 | FALSE |
| **17** | 73b81e30 | 46 | 83 | 0.554217 | FALSE |
| **18** | bbcf14e4 | 0 | 83 | 0 | FALSE |
| **19** | 096a6f32 | 0 | 82 | 0 | FALSE |
| **20** | 9af87478 | 3 | 82 | 0.036585 | TRUE |

At this point I tried to analyze if I could create a set of heterogeneous buckets to categorize each user. However I figured that the classification algorithms discussed below would implicitly utilize such information in building the model.

## Interaction between variables

When the above analysis did not yield any concrete results then I ventured to explore alternatives to identify strong predictive interactions between variables. In this section I explored four algorithms: Random forest, Naïve Bayes, Logistic Regression, Online Gradient Descent.

To begin with, below I present the output of finding buckets based on combination of app\_category, app\_domain and app\_id where CTR >90% and CTR>80%. We see that we have a few records that meet these criteria however such interactions can also be captured as part of the classification algorithms mentioned above.

> AppSummary[AppSummary$Freq>10 & AppSummary$CTR>0.9,]

app\_category app\_domain app\_id Clicks Freq CTR

47 07d7df22 7801e8d9 5054e8a9 24 26 0.9230769

501 0f2161f8 2347f47a 5fde3dd9 206 225 0.9155556

693 0f2161f8 2347f47a 95827a92 409 423 0.9669031

> AppSummary[AppSummary$Freq>10 & AppSummary$CTR>0.8,]

app\_category app\_domain app\_id Clicks Freq CTR

47 07d7df22 7801e8d9 5054e8a9 24 26 0.9230769

501 0f2161f8 2347f47a 5fde3dd9 206 225 0.9155556

693 0f2161f8 2347f47a 95827a92 409 423 0.9669031

1011 0f2161f8 2347f47a eeeac987 9 11 0.8181818

1101 0f2161f8 33da2e74 08a2b59c 59 67 0.8805970

2195 0f2161f8 7801e8d9 e732e74e 11 13 0.8461538

2800 0f2161f8 d9b5648e a065fb4f 25 30 0.8333333

3387 8ded1f7a 73fc6786 7a288b48 9 11 0.8181818

3741 cef3e649 7801e8d9 b9663f57 10 12 0.8333333

3985 f95efa07 2347f47a 9d1bad7b 45 55 0.8181818

4073 f95efa07 ae637522 5f8f6c12 66 76 0.8684211

### Random Forest

I conducted a first run of Random Forest in R to data mine through the training data and find the variable importance. Since the factors in R cannot take more than 53 categories in the variable I restricted my model to variables with less than 20 categories to further aid in obtaining a parsimonious model.

The Random Forest model did not have a good LogLoss measure but did provide the variable importance as mentioned below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | MeanDecreaseAccuracy | MeanDecreaseGini |
| C1 | -0.00901 | 0.017931 | -0.006294897 | 635.5799 |
| banner\_pos | -0.08638 | 0.125315 | -0.065073354 | 1110.8755 |
| site\_category | -0.10783 | 0.202011 | -0.076643769 | 5273.6782 |
| app\_category | -0.01215 | 0.067933 | -0.004086483 | 5528.7731 |
| device\_type | -0.01119 | 0.021351 | -0.007915865 | 544.8614 |
| device\_conn\_type | -0.00114 | 0.009038 | -0.000114158 | 2761.8045 |
| C15 | -0.00763 | 0.026379 | -0.004203265 | 2145.3359 |
| C16 | -0.00878 | 0.046226 | -0.003245993 | 4092.8487 |
| C18 | -0.03126 | 0.120042 | -0.016028939 | 6458.4209 |
| DayOfWeek | 0.002941 | 0.011007 | 0.003753228 | 1052.8729 |

As we see the *MeanDecreaseGini* is the most for C18 and then for app\_category and then site\_category. This was the motivation to chose app\_category for the “Feature Generation over Chronological Data”.

### Naïve Bayes

Next I attempted to build a classic Naïve Bayes classifier. The probability for each test case was computed by taking a product of the conditional probability of observing the category in each variable of that case given a particular response. Since I am taking the product, hence implying that the variables are independent, I excluded some “correlated” (read associated) variables (such as site\_id and site\_domain; C15 and C16) to ensure that such assumptions make sense.

Here is the model performance output:

table(OOSValSample$click, Prob$Prob1>Prob$Prob0)

FALSE TRUE

0 6626 1677

1 920 777

table(OOSValSample$click, Prob$Prob1>Prob$Prob0)/c(sum(OOSValSample$click==0),sum(OOSValSample$click))

FALSE TRUE

0 0.7980248 0.2019752

1 0.5421332 0.4578668

The logloss for Naïve Bayes was 0.71 which seems better than the Random Forest but was still far down on the leaderboard.

### Logistic Regression

I applied Logistic Regression on a dataset with 11 variables that have <=20 categories each. The results are summarized below. We can see that there is a big improvement in logloss to ~0.44.

Logistic

LogLoss(ModelDataV2$click, fit$fitted.values)

[1] 0.4396754

Call:

glm(formula = click ~ ., family = binomial, data = ModelDataV2)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.5033 -0.6659 -0.5641 -0.3845 3.0735

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 8.406e+01 5.565e+00 15.104 < 2e-16 \*\*\*

C1 -8.661e-02 5.543e-03 -15.625 < 2e-16 \*\*\*

banner\_pos 1.410e-01 4.447e-03 31.700 < 2e-16 \*\*\*

site\_category28905ebd 1.789e+00 1.423e-01 12.573 < 2e-16 \*\*\*

site\_category335d28a8 5.547e-01 1.477e-01 3.757 0.000172 \*\*\*

site\_category3e814130 1.573e+00 1.424e-01 11.046 < 2e-16 \*\*\*

site\_category42a36e14 1.088e+00 2.808e-01 3.876 0.000106 \*\*\*

site\_category50e219e0 1.444e+00 1.426e-01 10.122 < 2e-16 \*\*\*

site\_category5378d028 -6.518e+00 1.645e+01 -0.396 0.691908

site\_category70fb0e29 1.179e+00 1.754e-01 6.725 1.75e-11 \*\*\*

site\_category72722551 5.375e-01 1.714e-01 3.135 0.001716 \*\*

site\_category74073276 -6.506e+00 7.246e+01 -0.090 0.928462

site\_category75fa27f6 1.015e+00 1.459e-01 6.955 3.52e-12 \*\*\*

site\_category76b2941d -4.603e-01 1.592e-01 -2.891 0.003843 \*\*

site\_category8fd0aea4 9.304e-01 7.750e-01 1.200 0.229977

site\_category9ccfa2ea -2.536e-01 1.036e+00 -0.245 0.806539

site\_categorya818d37a -2.723e+00 1.013e+00 -2.687 0.007208 \*\*

site\_categorybcf865d9 -1.804e-01 5.332e-01 -0.338 0.735030

site\_categoryc0dd3be3 1.182e+00 1.567e-01 7.541 4.65e-14 \*\*\*

site\_categoryda34532e -5.943e+00 5.123e+01 -0.116 0.907655

site\_categorydedf689d 3.264e+00 1.554e-01 20.999 < 2e-16 \*\*\*

site\_categorye787de0e 8.722e-02 6.086e-01 0.143 0.886041

site\_categoryf028772b 1.409e+00 1.422e-01 9.908 < 2e-16 \*\*\*

site\_categoryf66779e6 -2.494e-01 1.485e-01 -1.679 0.093076 .

app\_category09481d60 -2.503e-03 4.673e-02 -0.054 0.957284

app\_category0bfbc358 -2.013e+00 1.023e+00 -1.968 0.049077 \*

app\_category0d82db25 -8.247e+00 4.184e+01 -0.197 0.843721

app\_category0f2161f8 -4.137e-01 1.277e-02 -32.405 < 2e-16 \*\*\*

app\_category0f9a328c -6.731e-02 1.546e-01 -0.435 0.663229

app\_category18b1e0be 1.239e-01 7.694e-01 0.161 0.872070

app\_category2281a340 -2.424e+00 5.840e-01 -4.152 3.30e-05 \*\*\*

app\_category2fc4f2aa -7.198e-01 1.035e+00 -0.696 0.486609

app\_category4681bb9d -2.320e-01 1.605e-01 -1.445 0.148461

app\_category4ce2e9fc 8.239e-02 8.736e-02 0.943 0.345599

app\_category5326cf99 -8.090e+00 2.286e+01 -0.354 0.723386

app\_category7113d72a -8.333e+00 1.705e+01 -0.489 0.624977

app\_category75d80bbe -5.447e-01 7.250e-02 -7.513 5.77e-14 \*\*\*

app\_category79f0b860 -3.660e-01 4.833e-01 -0.757 0.448800

app\_category86c1a5a3 -7.963e+00 7.246e+01 -0.110 0.912498

app\_category879c24eb -4.217e-01 1.108e-01 -3.805 0.000142 \*\*\*

app\_category8ded1f7a -5.084e-01 1.751e-02 -29.042 < 2e-16 \*\*\*

app\_category8df2e842 1.117e+00 2.168e-01 5.154 2.54e-07 \*\*\*

app\_categorya3c42688 -8.168e-01 1.790e-01 -4.564 5.02e-06 \*\*\*

app\_categorya7fd01ec -1.272e-02 7.593e-01 -0.017 0.986639

app\_categorya86a3e89 -9.666e-02 2.485e-01 -0.389 0.697281

app\_categorybd41f328 -6.934e+00 7.246e+01 -0.096 0.923768

app\_categorybf8ac856 -8.180e+00 7.246e+01 -0.113 0.910123

app\_categorycef3e649 -5.318e-01 1.577e-02 -33.722 < 2e-16 \*\*\*

app\_categoryd1327cf5 -9.335e-02 3.687e-02 -2.532 0.011335 \*

app\_categorydc97ec06 1.781e-02 4.911e-02 0.363 0.716868

app\_categoryf95efa07 3.960e-01 1.623e-02 24.399 < 2e-16 \*\*\*

app\_categoryfc6fa53d -1.098e+00 1.181e-01 -9.301 < 2e-16 \*\*\*

device\_type -3.859e-03 1.101e-02 -0.350 0.726010

device\_conn\_type -2.041e-01 3.014e-03 -67.740 < 2e-16 \*\*\*

C15 -1.274e-03 1.066e-04 -11.954 < 2e-16 \*\*\*

C16 5.160e-03 3.489e-05 147.902 < 2e-16 \*\*\*

C18 6.374e-02 1.534e-03 41.560 < 2e-16 \*\*\*

DayOfWeekMon 9.660e-03 7.923e-03 1.219 0.222753

DayOfWeekSat 2.614e-02 7.813e-03 3.346 0.000821 \*\*\*

DayOfWeekSun 8.838e-03 7.580e-03 1.166 0.243596

DayOfWeekThu 6.159e-02 7.554e-03 8.153 3.56e-16 \*\*\*

DayOfWeekTue -4.152e-02 6.566e-03 -6.324 2.55e-10 \*\*\*

DayOfWeekWed -5.473e-02 7.296e-03 -7.502 6.31e-14 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2075473 on 2266416 degrees of freedom

Residual deviance: 1992975 on 2266354 degrees of freedom

AIC: 1993101

Number of Fisher Scoring iterations: 8

Due to the limited time, further work was not done to polish the model above. Ideally one would remove the insignificant terms and identify the most significant categories to be used. Further I would have tried all variable combinations to assess the best combination.

### Online Gradient descent

Perhaps one of the best ways to make use of all training data is by using Gradient Descent. This technique can be implemented in batch processing mode as the model learns with each input and can be implemented to provide predictions on the fly.

Gradient descent helped to bring down the logloss to ~0.394. The code for gradient descent can be found here: 

# Conclusion

The problem is very unique since the data only contains categorical values. Hence I believe the key is to find different characteristic segments that behave differently and build classification models on them. Association analysis can prove useful in identifying these segments as it can help identifying pairs of variables that separate each other’s characteristics.

Upon a cursory exploration of feature engineering techniques no robust technique was found. After applying several classification algorithms it seems that online gradient descent is the best performing.

The R code for the analysis above is attached here: 

# Further work

* Also explore neural networks to identify interactions.
* Conduct a more robust analysis focusing on the common categories between Train and Test
* Obtain pockets of data that have either too high or too small CTR and build separate models to capture the different characteristic.
* Explore the Feature generation analysis above based on all sets of combination of the variables, beyond the cursory analysis, to get a more complete picture.
  + Derive bagging and counting features grouped by different variables and explore their predictive power in the classification algorithms
* Explore regularization for OGD.
  + Also explore running OGD for different sorting order and maybe for a subset of variable or subset of categories or even over a subset of sample.