Predicting Which Recommended Content Users Click

Pramod Varma (github@**pramodv79**)

# Abstract

*In this paper, I explore application of data science to build an advertisement prediction engine. Using curated data provided Outbrain I provide my findings on data exploration, feature selection, model selection, prediction and evaluation. I compare two machine learning techniques and provide results from them. Finally, I conclude with learnings and future work for this analysis.*

# Introduction

The internet is a stimulating treasure trove of possibility. Every day we stumble on news stories relevant to our communities or experience the serendipity of finding an article covering our next travel destination. [Outbrain](http://www.outbrain.com/), the web’s leading content discovery platform, delivers these moments while we surf our favorite sites. Currently, Outbrain pairs relevant content with curious readers in about 250 billion personalized recommendations every month across many thousands of sites. In this paper, I explore to predict which pieces of content outbrain’s global base of users are likely to click on. Improving Outbrain’s recommendation algorithm will mean more users uncover stories that satisfy their individual tastes.[1]

# Data Exploration

The dataset for this project was provided by Outbrain as part of a Kaggle competition. It contains a sample of users’ page views and clicks, as observed on multiple publisher sites in the United States between 14-June-2016 and 28-June-2016. Each viewed page or clicked recommendation is further accompanied by some semantic attributes of those documents. For full details, see data specifications below.

The dataset contains numerous sets of content recommendations served to a specific user in a specific context. Each context (i.e. a set of recommendations) is given a display\_id. In each such set, the user has clicked on at least one recommendation. The identities of the clicked recommendations in the test set are not revealed. The task is to rank the recommendations in each group by decreasing predicted likelihood of being clicked.

## Exploration and Pre-processing of data

Outbrain provided total of eight datasets as described below:

1. page\_views: Describes features of all viewed pages, regardless of an advertisement being clicked.
2. events: Consists of features of pages viewed when one displayed advertisement was clicked.
3. promoted content: provides information on advertised content.
4. clicks\_train/clicks\_test: provides examples with labels to be used for training and examples without labels to be used for testing.
5. documents\_meta: describes documents’ metadata.
6. documents\_entities, 7) documents\_topic, and 8) documents\_categories: provide mentioned entities (person, place, or location), topic, and taxonomy of categories of the documents, respectively.

Since the data was normalized into separate datasets, it was necessary to join them to form a de-normalized fact table. I decided to load all the individual raw files into a PostgreSQL DB locally. After loading all the data, the size of the db was close to 9GB. This did not include full set of page\_view log file which was uncompressed about 100GB.

 table         | table\_size | related\_objects\_size | total\_table\_size | live\_rows

----------------------+------------+----------------------+------------------+-----------

 promoted\_content     | 32 MB      | 32 kB                | 32 MB            |    559583

 documents\_meta       | 164 MB     | 64 kB                | 164 MB           |   2999334

 documents\_categories | 273 MB     | 88 kB                | 273 MB           |   5481471

 documents\_entities   | 446 MB     | 136 kB               | 446 MB           |   5537633

 page\_views\_sample    | 726 MB     | 200 kB               | 726 MB           |  10000008

 documents\_topics     | 564 MB     | 160 kB               | 564 MB           |  11325980

 events               | 1822 MB    | 480 kB               | 1822 MB          |  23121858

 clicks\_test          | 1361 MB    | 360 kB               | 1361 MB          |  32225335

 clicks\_train         | 4336 MB    | 1112 kB              | 4337 MB          |  87141751

 TOTAL                | 9724 MB    | 2632 kB              | 9726 MB          | 178392953

Figure 1

Due to the size of the data and scope the exercise to be completed timely, I decided to exclude examples found in Page Views (2B of rows) and only consider the 87 million examples contained in Events. Events contains the examples of page views that resulted in a click for one of the featured advertisements contain useful information to make click predictions.

After exploring each of the data set individually (except page\_view), following information was learnt from the data sets (details of this exploration are provided in notebook [2]).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dataset** | **Column** | **Cardinality** | **Missing** | **%Missing** | **Total Rows** |
| clicks\_train | display\_id | 16,874,593 | 0 | 0.00000 | 87,141,731 |
| clicks\_train | ad\_id | 478,950 | 0 | 0.00000 | 87,141,731 |
| clicks\_train | clicked | 2 | 0 | 0.00000 | 87,141,731 |
|  |  |  |  |  |  |
| events | platform | 4 | 0 | 0.00000 | 23,120,126 |
| events | display\_id | 23,120,126 | 0 | 0.00000 | 23,120,126 |
| events | document\_id | 894,060 | 0 | 0.00000 | 23,120,126 |
| events | uuid | 19,794,967 | 0 | 0.00000 | 23,120,126 |
| events | geo\_location | 2,989 | 340 | 0.00001 | 23,120,126 |
| events | timestamp | 22,896,622 | 0 | 0.00000 | 23,120,126 |
|  |  |  |  |  |  |
| document\_meta | document\_id |  |  |  | 2,999,334 |
| document\_meta | source\_id | 14,395 | 2,518 | 0.00084 | 2,999,334 |
| document\_meta | publisher\_id | 1,260 | 64,024 | 0.02135 | 2,999,334 |
| document\_meta | publish\_time | 49,914 | 1,011,118 | 0.33711 | 2,999,334 |
|  |  |  |  |  |  |
| document\_topics | document\_id | 2,495,423 | 0 | 0.00000 | 11,325,960 |
| document\_topics | topic\_id | 300 | 0 | 0.00000 | 11,325,960 |
| document\_topics | confidence\_level | 10,124,758 | 0 | 0.00000 | 11,325,960 |
|  |  |  |  |  |  |
| documents\_categories | document\_id | 2,828,649 | 0 | 0.00000 | 5,481,475 |
| documents\_categories | category\_id | 97 | 0 | 0.00000 | 5,481,475 |
| documents\_categories | confidence\_level | 21,900 | 0 | 0.00000 | 5,481,475 |
|  |  |  |  |  |  |
| documents\_entities | document\_id | 1,791,420 | 0 | 0.00000 | 5,537,552 |
| documents\_entities | entity\_id | 1,326,009 | 0 | 0.00000 | 5,537,552 |
| documents\_entities | confidence\_level | 2,678,533 | 0 | 0.00000 | 5,537,552 |
|  |  |  |  |  |  |
| promoted\_content | document\_id | 185,709 | 0 | 0.00000 | 559,583 |
| promoted\_content | ad\_id | 559,583 | 0 | 0.00000 | 559,583 |
| promoted\_content | campaign\_id | 34,675 | 0 | 0.00000 | 559,583 |
| promoted\_content | advertiser\_id | 4,385 | 0 | 0.00000 | 559,583 |

Figure 2

Given the amount of data was large I decided to only picked categories, topics and entities of the document with highest confidence\_level for first iteration. Additional tables were created. Below are the intermediate tables created using sql. SQL Code used could be found at [3]

 table        | table\_size | related\_objects\_size | total\_table\_size | live\_rows

--------------------------+------------+----------------------+------------------+-----------

 clicks\_train\_fact\_limit  | 169 MB     | 8192 bytes           | 169 MB           |   1000000

 documents\_entities\_top   | 144 MB     | 38 MB                | 183 MB           |   1791420

 documents\_topics\_top     | 124 MB     | 53 MB                | 178 MB           |   2495423

 documents\_categories\_top | 141 MB     | 61 MB                | 201 MB           |   2828649

 clicks\_test\_fact         | 5247 MB    | 2071 MB              | 7318 MB          |  32224999

 clicks\_train\_fact        | 14 GB      | 5600 MB              | 20 GB            |  87147369

Figure 3

# Feature Selection

For the first pass columns highlighted in green were considered as features. Columns highlighted in red were not considered. Also, since I am considering only the top meta data with high confidence I excluded the confidence level value. Cardinality of uuid was very high to provide any meaningful information. Since I was not planning to do time series modeling timestamp and publish\_timestamp was ignored. Below were the 12 features that were used for the training the model.

|  |  |
| --- | --- |
| **Category** | **Features** |
| User | uuid, geo\_location, platform, timestamp |
| Display | display\_id, document\_id, source\_id, publisher\_id, publish\_time, topic\_id, (topic)confidence, category\_id, (category)confidence\_level, entity\_id, (entity)confidence\_level |
| Ad | ad\_id, document\_id, campaign\_id, advertiser\_id |

Figure 4

# Deciding on estimator

Response variable is a 0 and 1 indicating if an ad is clicked or not clicked which make us believe it is a classification problem but the outbrain submission file is expecting the ads to be ranked by probability due to which it becomes a prediction problem. I plan to use a classification model for the purpose of this exercise and calculation of probability of the response variable to rank the ads is deferred in the later stage as future work.

I decided to use the random forest estimator. The primary reason was to develop ensemble of trees that can learn different behaviors from the feature set. By randomly leaving out candidate features from each split, **Random Forests "decorrelates" the trees,** such that the averaging process can reduce the variance of the resulting model. For comparison sake, I build a Generalized Linear Model as a second model.

# Model Training

After I created a de-normalized fact tables with above feature the size of the table is ~20GB which is unmanageable on local computer. To move forward I reduced the scope of the dataset to first 1M out of 87M (> 10% of the training population).

For ease of iteration I explored a data science environment H20. Once the data is loaded into the H2O data frame it provides easy way to change the data type, build models, predict and compute score. Since most of the features were categorical H20’s feature to convert them to enum was used.

# MODEL 1: Random Forest

For Random Forest as the estimator using Train, Test and Validate splits of 70%, 20% and 10% respectively. Key parameters that were selected

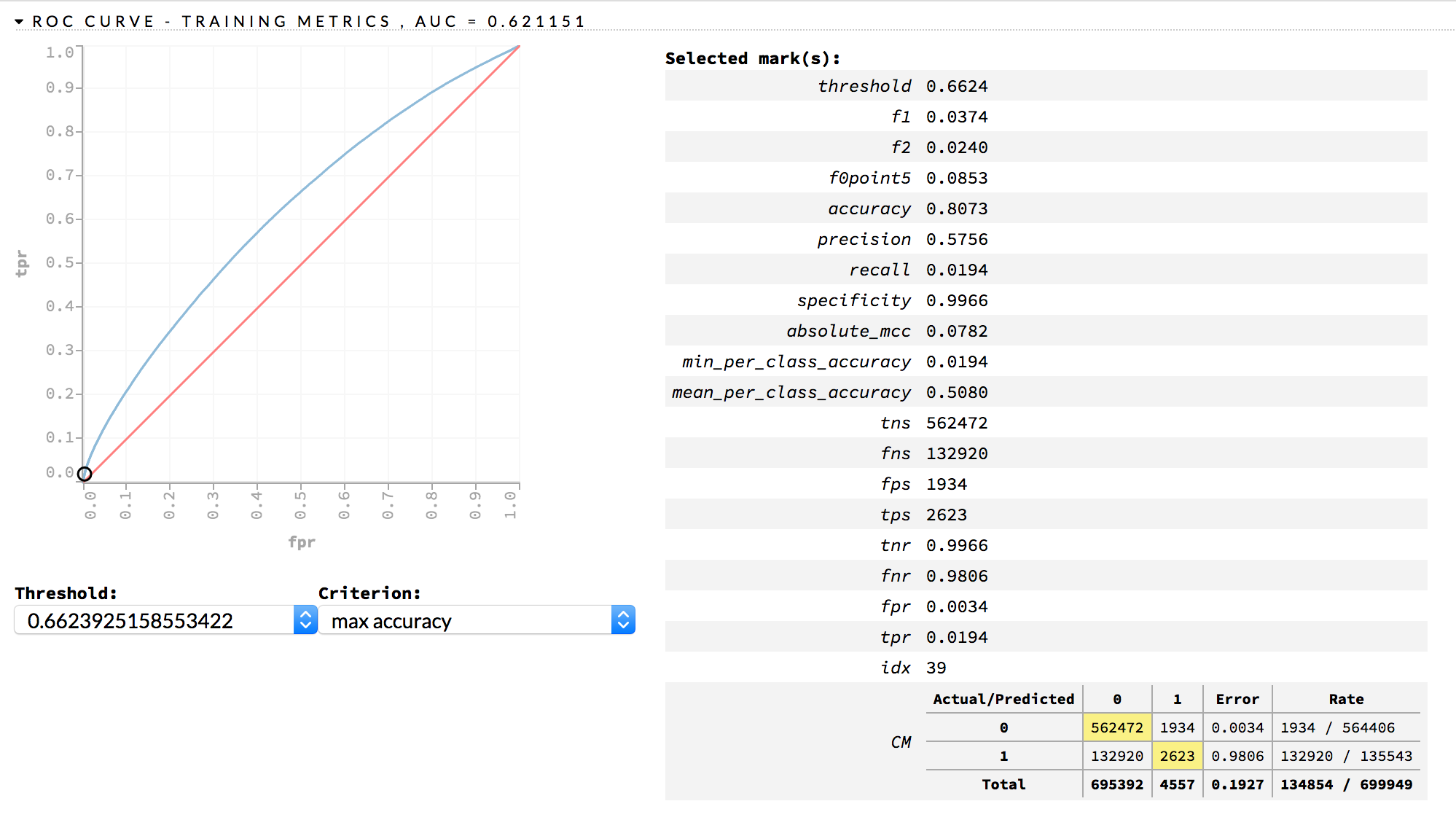
1. ntrees = 50
2. depth = 20

## Model Training

While training model it was observed that the model learns quite rapidly until 15 trees and later on learns very slowly. For faster performance, we could reduce the number of trees to only 15 trees. Accuracy Lift as compared to training lift is quite low after 15 trees

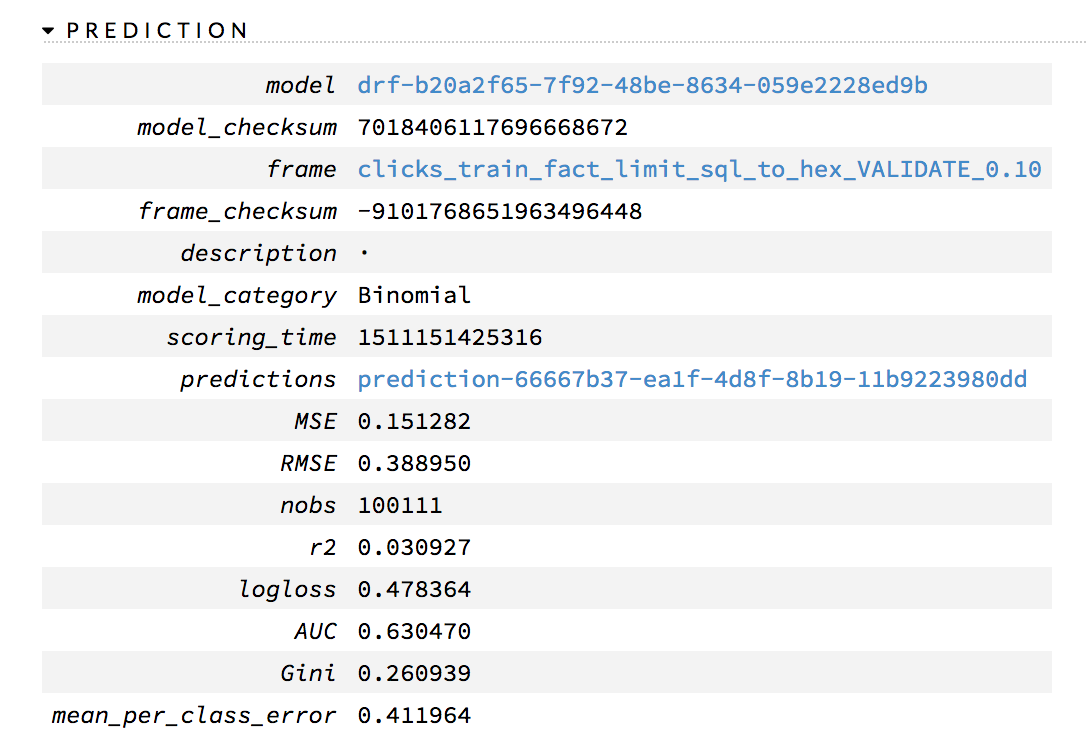
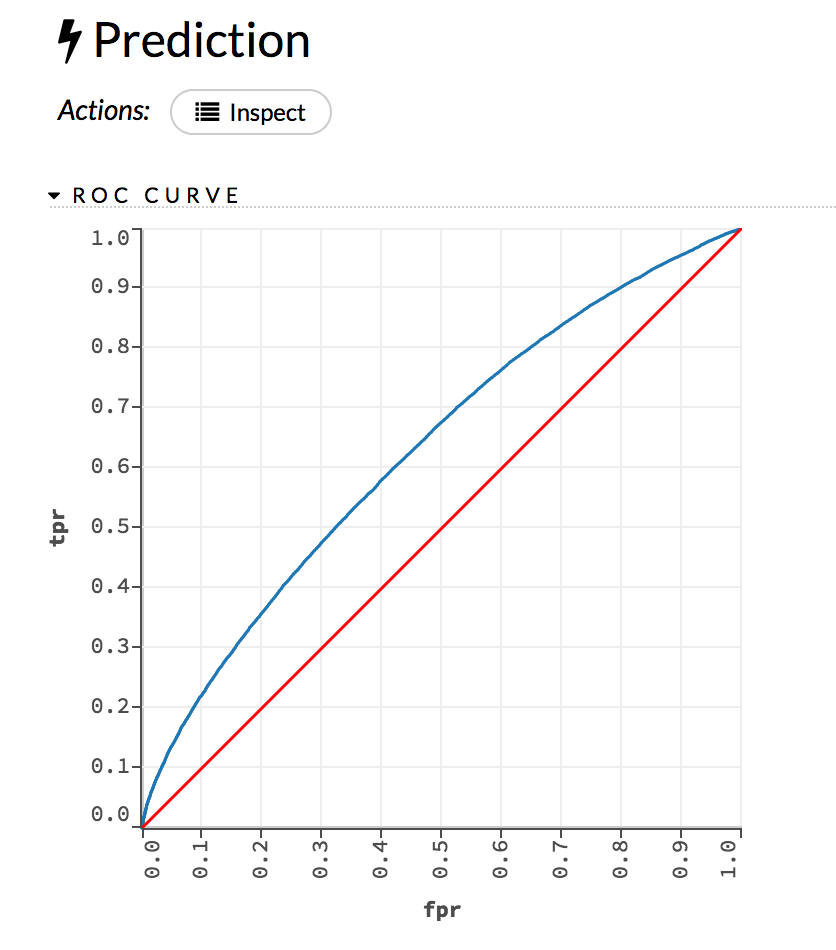


# After training the model ROC curve was generated with AUC 0.62. MSE was 0.153. Accuracy is 0.81



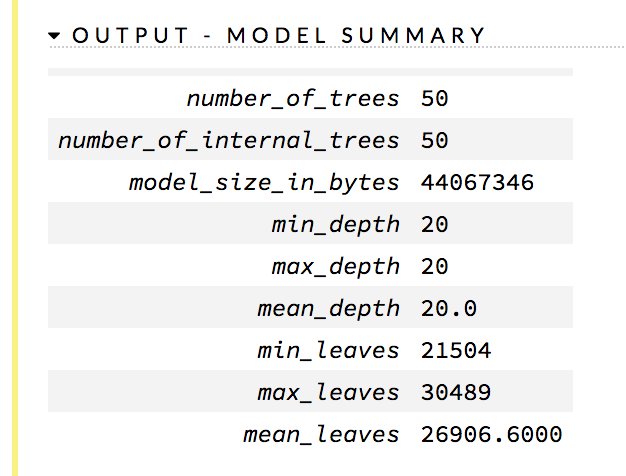
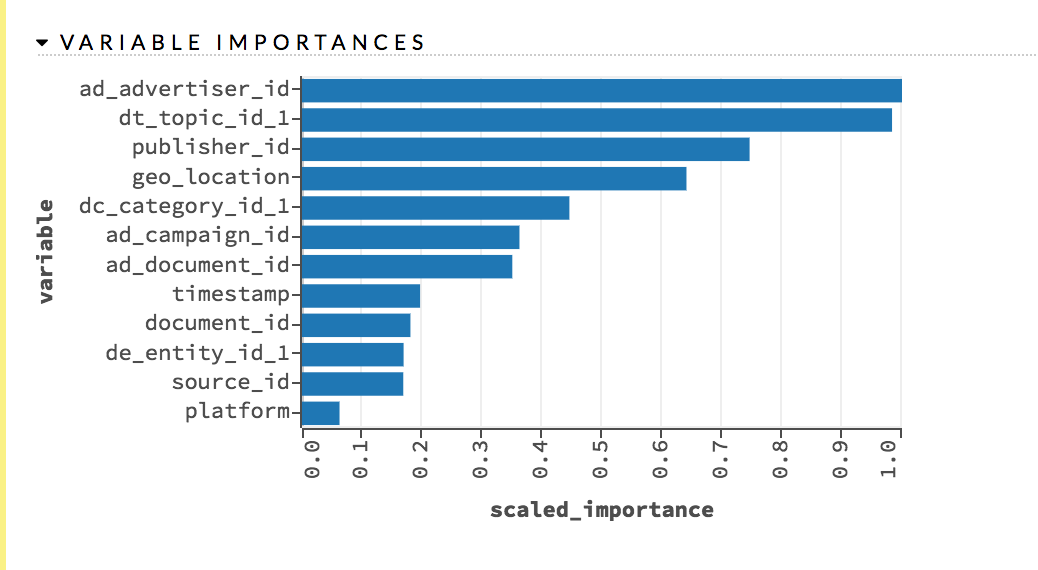
# 

## Model Prediction



## Variable Importance

After model was build and tested I evaluated important variables. And below is the list that surfaced along with importance

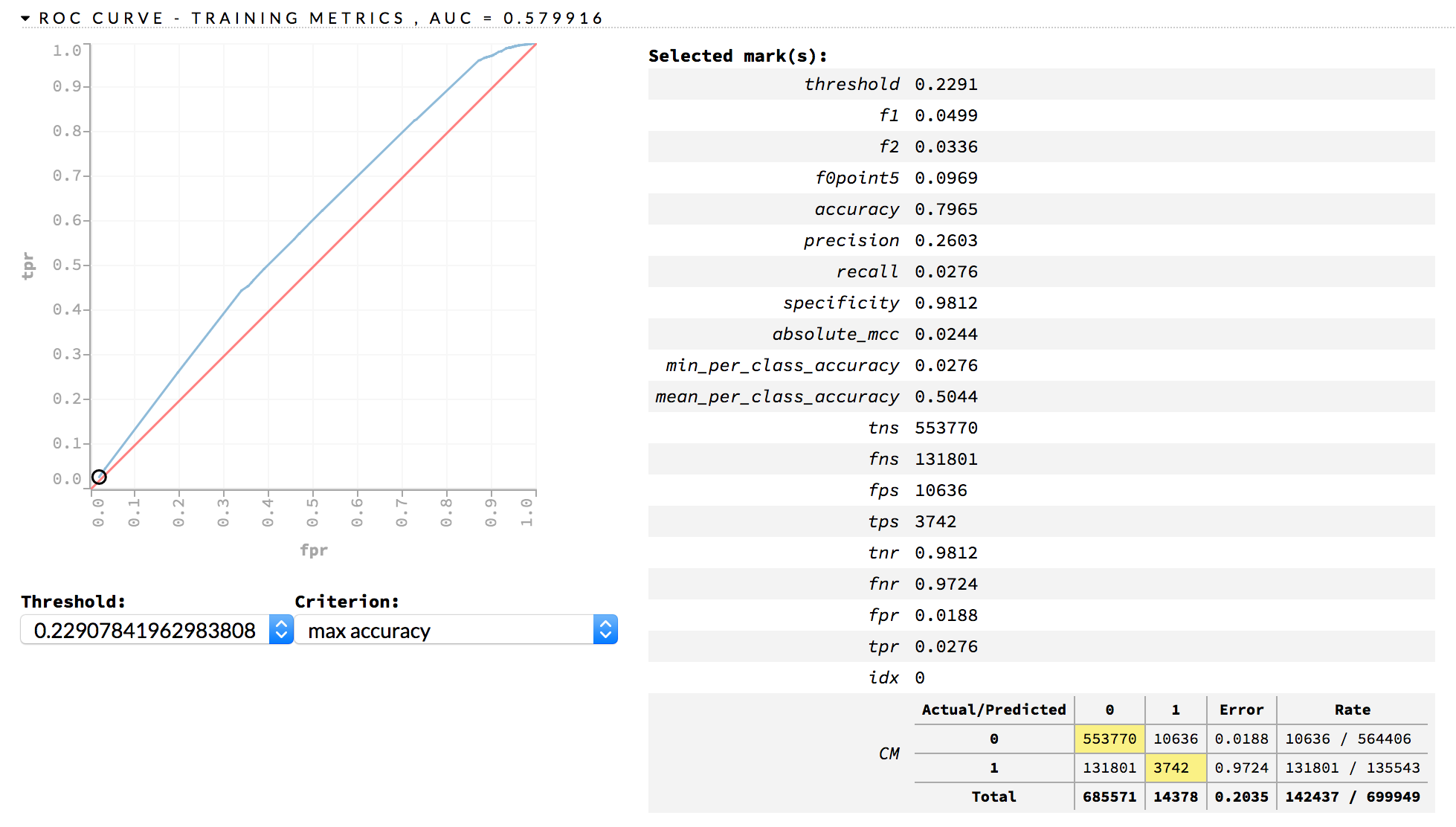


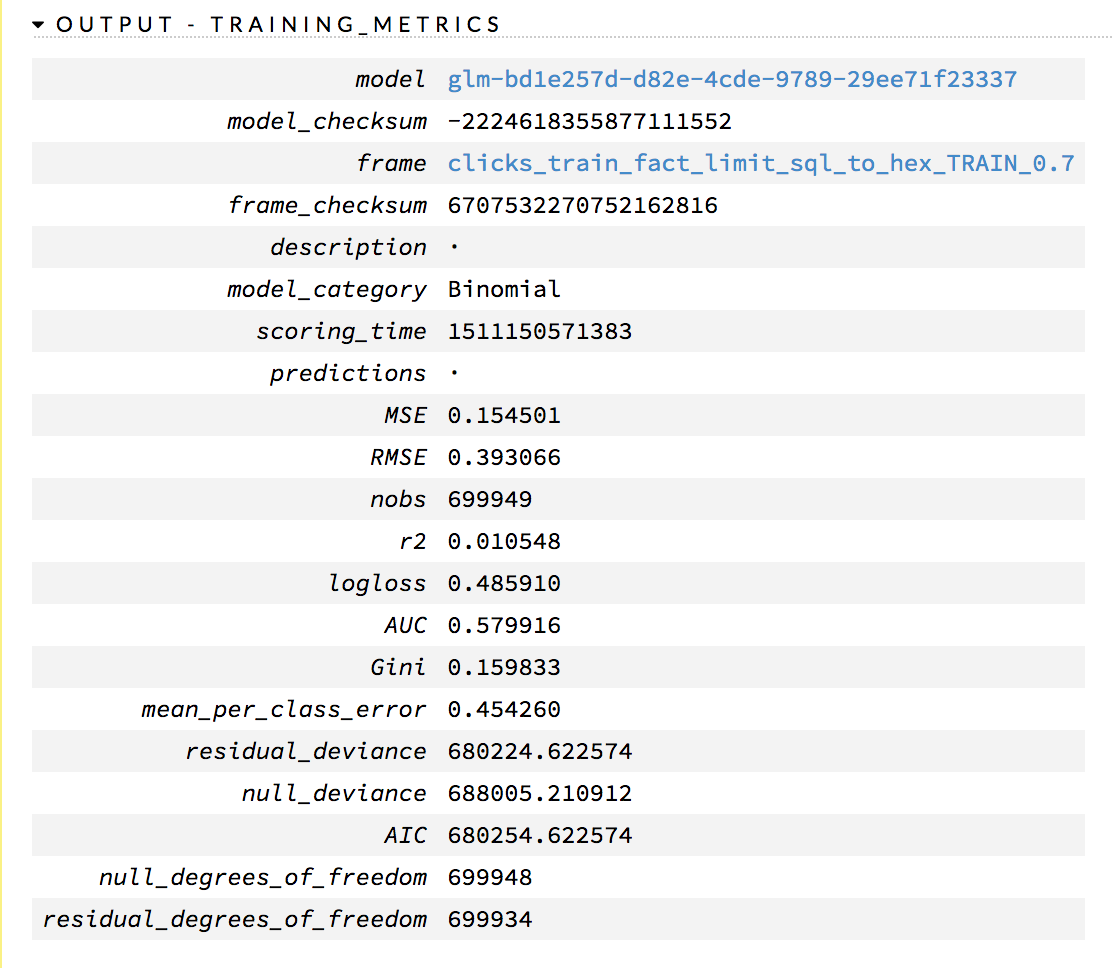
# Model 2: Generalized Linear Model

As a comparison, I also built another model Generalized Linear Model using the same data

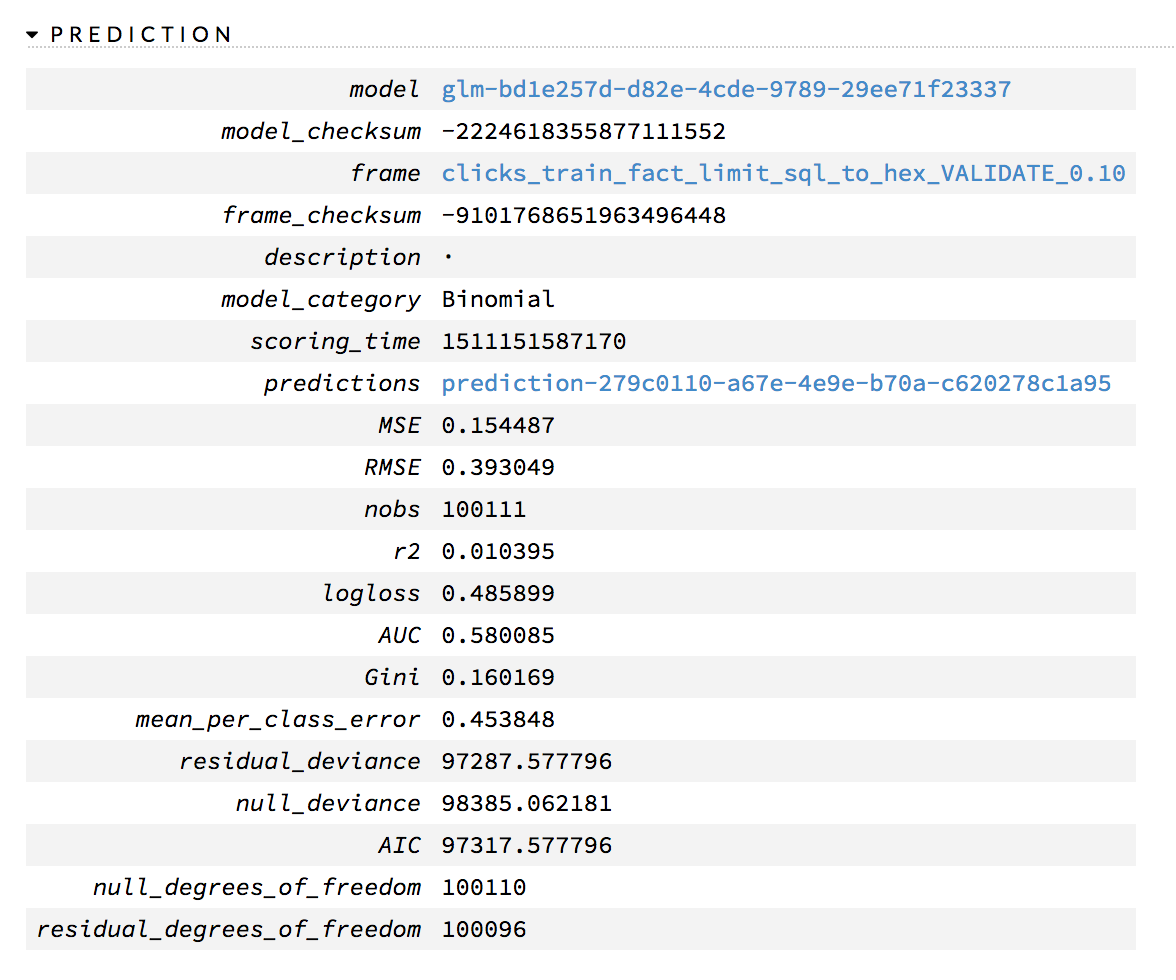
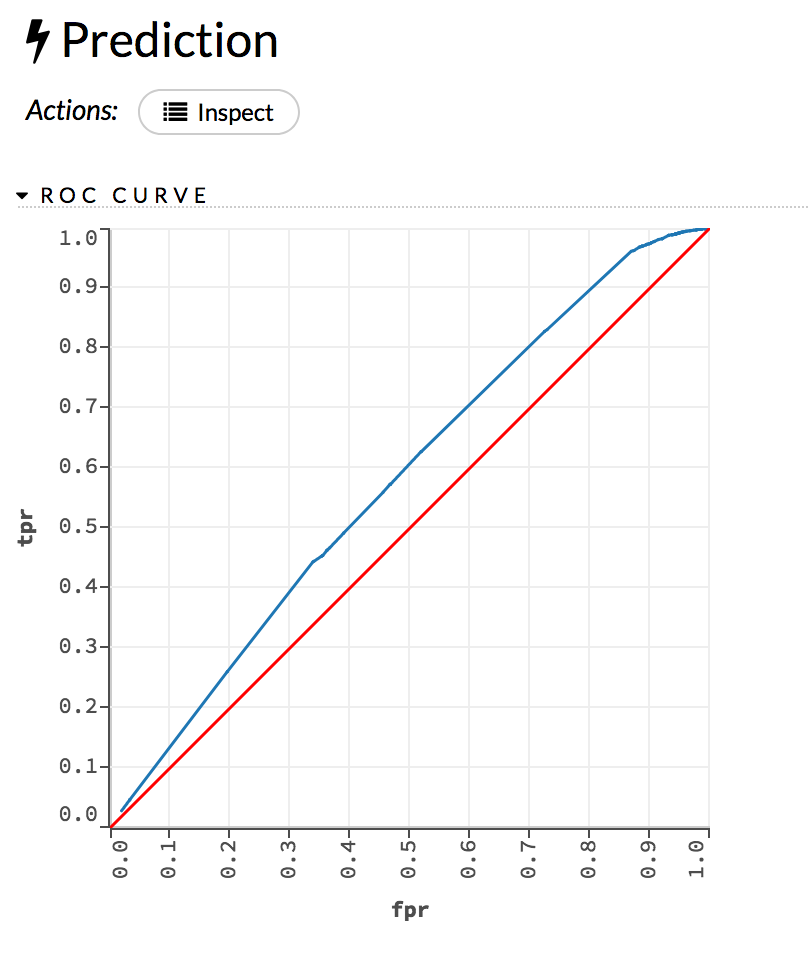
## Model Training

Observed that AUC is lower than random forest and also higher (0.393). Accuracy is 0.7965

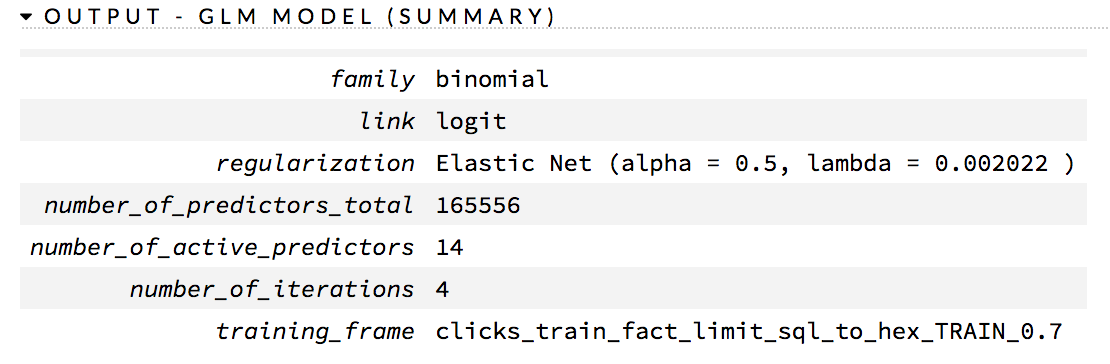
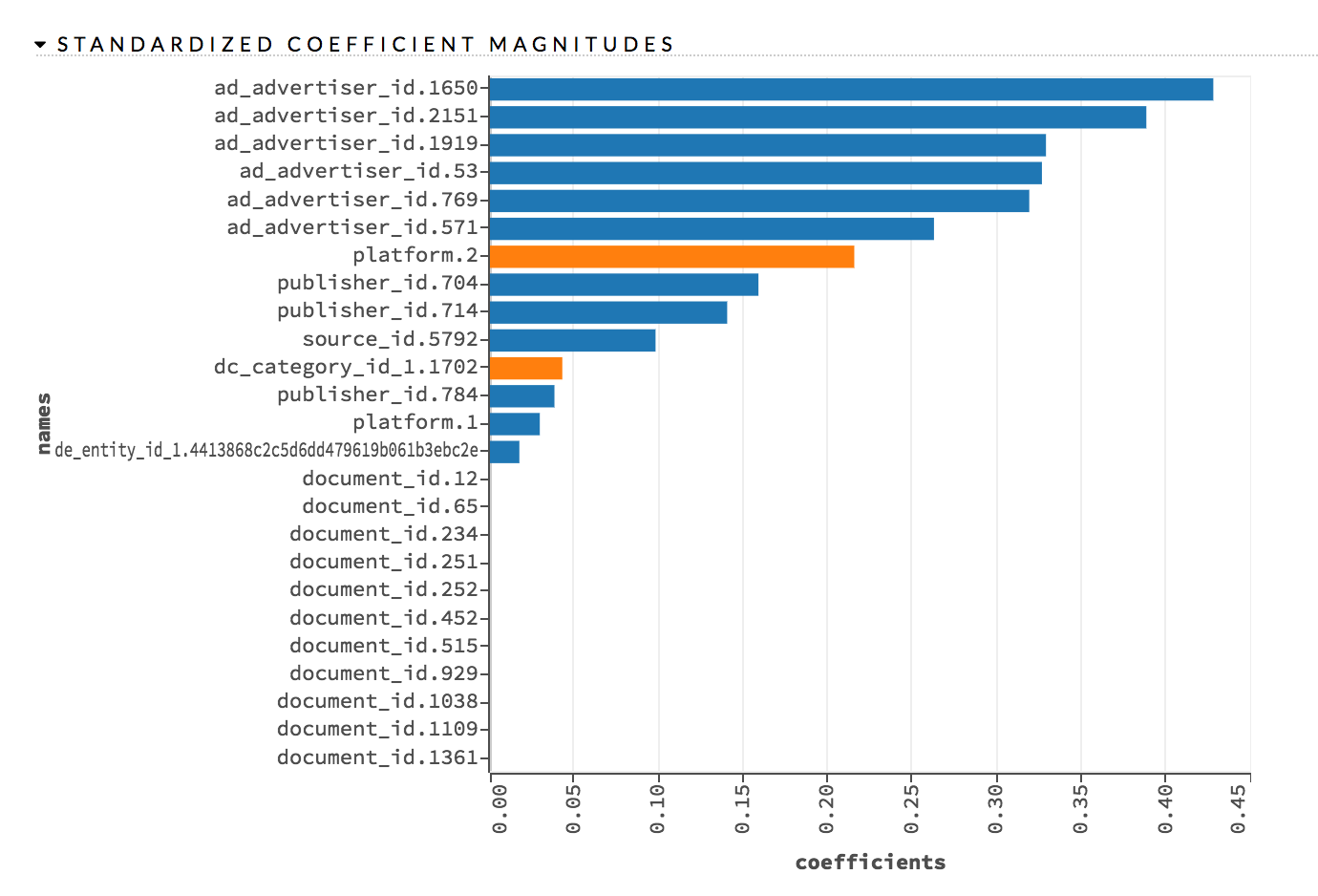




## Model Prediction



## Variable Importance



# Challenges and Success stories

Since Outbrain has masked the actual document data all we have are categorical ids. Expanding all these categorical variables results in a very high number of features. For generalized Linear Model the number of variables exploded to 165K.

Using H2O to run through feature selection, data cleaning, model building was quick as compared to doing steps in python.

# Conclusion and Learnings

Random forest model performed slightly better than Linear Model. Most difficult part is understanding the data and perform the leaning work.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | |  | | |

# Future Work

Due to reduced time for the submission, I was able to only develop only one model for random forest and linear model. As part of future work it will be beneficial to do following

* Tune random forest further for number of depth, estimator and feature selection
* Add additional features to the model to improve accuracy
  + Pick top 3 topics, categories and entities by confidence level instead of one
  + Enrich document ID features for the Ad
  + Cluster high cardinality features into low cardinality features using DB scan
* Use other machine learning models to improve accuracy (e.g. Gradient boosting)
* Move to cloud to run on a larger training set and see if page\_views would be beneficial

# Reference

[1] <https://www.kaggle.com/c/outbrain-click-prediction/data>

[2] Python notebook: <https://github.com/Pramodv79/DSClickPrediction/blob/master/notebooks/EDA_OutbrainRawDataset.ipynb>

[3] Text file with SQL Commands to create FACT table for model building

<https://github.com/Pramodv79/DSClickPrediction/blob/master/database/SQL_CREATE_CLEAN_TABLES>

[4] Python notebook used to load the dataset into H2O

[ insert link to H20 flow notebook]

[5] H20 notebook used to train, validate and predict Random Forest and GLM model

[ insert link to H20 flow notebook]