UCL RTB CTR Estimation Challenge 2016

Technical report

Group Name: LongInt

Group members: Ahmed Awad (15101640), Betran Jacob(15040593), Manal Alonaizan(14077649)

1. Introduction

The Internet has become the most widely used medium for advertising; a result of the ease by which an advertiser can target consumers who are likely to purchase their product and the speed at which a company can react to market changes. Consequently, several search engines and ad hosting sites offer a platform through which anyone can attempt to push advertisements to end users. However, in an effort to augment revenue, an appropriate ad needs to be selected such that the probability the user clicks the ad—upon which payment is made by the advertiser—is maximized.

The goal of this work is to tackle precisely the aforementioned problem: given a selection of user attributes, the features of the advertisements, and application context —such as web browser used, domain and URL— we make a prediction on the likelihood that the user will click an advertisement through the use of machine learning techniques. The rest of the report is organized as follows: Section 2 provides an overview of the statistical analysis we conducted to decide the features we used to train our model, Section 3 compares between a few machine learning algorithms that we tested in an attempt to solve this problem, Section 5 presents an evaluation of the tested algorithms, on the data-set provided, and finally Section 6 provides concluding statements.

2. Statistical Analysis and Feature Selection

We first performed a brief statistical analysis on the data to determine which features had a chance of being good predictors of whether a user clicks an ad or not. This is the first step before the training phase. The data-set provided for the challenge contains the following 23 features:

Features Listed as a tuple of ('Name', Data type):

1.) 'Weekday', int 2.) 'Hour', int 3.) 'Timestamp', int 4.) 'LogType', int 5.) 'UserID', str
6.) 'User-Agent', str 7.) 'IP', str 8.) 'Region', int 9.) 'City', int 10.) 'AdExchange', int
11.) 'Domain', str 12.)'URL', str 13.)'AnonyURL', str 14.) 'AdSlotID', str 15.) 'width', int
16.) 'Height', int 17. 'Visibility', int 18.) 'format', int 19.) 'floorPrice', int
20.) 'CreativeID', str 21.) 'KeyURL', str 22.) 'Advetiser ID', int 23.) 'UserTags', str

We classified all the features as categorical to simplify the programming. One of the many ways by which categorical variables can be represented for data analysis is one-hot encoding, discussed below.

One-hot Encoding:

A one-of-K/one-hot encoding has been performed on all categorical features. The reason behind this follows from the fact that classification algorithms work directly on numerical data. As a result, we need a

way to represent strings, and also numbers that are inherently categories - such as weekday and hour from the dataset- in a manner that, not only maintains the information presented by a feature, but also does not allow the algorithm to infer ordering relationships that do not exist (the case when representing strings directly as numbers). Hence, we convert all categorical features using one-hot encoding. The encoding transforms each categorical feature to a list of additional features based on the unique values within the feature. The new features will have value set to 1 whenever it's active in the data set. We used the sklearn utility function **DictVectorizer()** to perform one-hot encoding. Figure 1 exemplifies how **one-hot** encoding splits a feature based on its unique values.

ID	Click	User_Agent
1	0	windows_ie
2	0	mac_safari
3	1	windows_ie
4	0	windows_ie

→

ID	Click	User_Agent	
		windows_ie	mac_safari
1	0	1	0
2	0	0	1
3	1	1	0
4	0	1	0

Figure 1: Before and after one-hot encoding

Feature Elimination:

Next, we carried out our statistical analysis, and eliminated features from the training data that are less useful to predict the probability a user will click an advertisement. To examine the correlation between a particular feature and the target variable ('click'), we analyzed the CTR (total clicks/number of entries) of each unique value taken by that feature. This was accomplished by grouping every unique value of a feature separately and calculating the mean of the 'click' column for each group.

The first set of eliminations were carried out on features that seemed to have a low influence on CTR. This is gleaned from calculating the CTR for each value of the feature, and concluding that the CTR does not vary much with the values of this feature. Examples of these calculations can be seen in Figure 2.

1.) Ad SlotID
2.) Timestamp
3.) UserID
4.) IP
5.) URL
6.) Key Page URL

Finally, we eliminated the features that have only one unique value since they do not aid in forming a prediction. The features eliminated through this process were:

1.) LogType
2.) Anonymous URL
3.) Advertiser ID

Feature Selection:

After the eliminations above, we examined the correlation of each feature with whether or not the ad was clicked. This has been done by evaluating the average number of times an ad was clicked in the total data-set and comparing it against the average number of times an ad was clicked when a feature is present as explained before. The average for the entire data-set, or CTR, is 0.000728; so any features which when present had a significantly higher CTR might be useful. Figure 2 shows some examples of features that have higher CTR values - especially when taking certain unique values within the feature. Below we provide some of the reasoning we applied in reducing the feature space to the features we selected.

- *User-Agent*

This feature indicates the browser and operating system used by a user when visiting a website. It has 10 unique values from the full data set. Six of the browsers have no click rates, but, we observed that when the user is using Google Chrome they are far more likely to click on an ad. Moreover, the usage of the three other browsers seem to have some correlation with whether or not an ad is clicked. This variance of one high, three medium and six nil influences on CTR is a good indication that this is a useful feature for attempting to predict whether or not a user clicks an ad.

- *Ad slot format and Ad Exchange*

As illustrated in the Figure 2 below, these features have **three unique** values each. Two of them do not seem to correlate with whether or not a user clicks an ad, however, a large majority of clicks occurred when the value of *Ad Slot format* is 5 and the when the value of *Ad Exchange* is 1. As a result, we use these features to form our prediction.

- *Ad slot Visibility*

The visibility feature has **four** unique values. Two of these values had higher CTR rate and the remaining two's contribution is comparatively very low. This variance could aid predictor to make some good calculations.

- *Ad slot Floor Price*

The *'Ad slot floor price'* feature has **ten** unique values. Observe from the table in Figure 2 that it seems like the majority of the ad clicks occurred when the '*Ad slot floor price'* was 118, and almost no ads were clicked when the floor price was some other value. Based on this, we concluded that we can use the *'Ad slot Floor Price'* as a feature.

- *Domain*

Intuitively, a user is more likely to click advertisements on certain websites, However, this feature had ~76000 unique values, and as a result it was difficult to conclude whether it would be a useful feature or not. After testing the predictor with and without the feature, we noted an increase in the average AUC with the '*Domain'* feature included, and therefore decided to use it as a feature.

User-Agent	CTR
linux_firefox	0.0
windows_opera	0.000362450163103
mac_safari	0.00493539346742
other_firefox	0.0
android_chrome	0.0108303249097
android_ie	0.0
mac_maxthon	0.0
windows_ie	0.000672465780169
android_firefox	0.0
mac_firefox	0.0

Ad slot format	CTR
0	0.000561364179269
5	0.00997112737528
1	0.00086871623392

Ad Exchange	CTR
2	0.000544520970439
3	0.000574171656299
1	0.00127279935376

Ad slot visibility	CTR
2	0.00042412373941
255	0.0012573584043
0	0.000645349346314
1	0.00163883898031

Ad slot Floor Price	CTR
118	0.000684931506849
232	0.0
13	0.0
60	0.0
187	0.0
132	0.0
136	0.0
116	0.0
160	0.0
105	0.0

Figure 2: CTR of the selected features

Feature Engineering

Further improvements can be achieved through feature engineering, an iterative process in which we build new features from existing ones. This is usually done through domain knowledge, but we instead rely on intuition to determine which features may be good predictors.

- *Weekday & Hour*

The time of day, and which day of the week may be good predictors of whether or not someone is going to click an ad. This information could be extracted from processing the timestamp. However, the '*WeekDay'* feature and '*Hour'* feature provide this information in the form of integers.

- *Ad Area*

The size of an ad seems to be a good predictor from the analysis illustrated above on '*Ad Slot Width'* and '*Ad Slot Height'*. We formed an additional feature '*Ad Area'* by calculating the area of the ad slot.

- *User Tags*

The attributes associated with a user may be a good indicator of whether or not they are likely to click an advertisement if presented in a certain context. The *'User Tags'* column in the dataset was a list of numbers. We iterated through the entire dataset and came up with a list of the unique tags present in the column. We then used each one as a new column wherein each entry was a '1' if the tag was present in the *'User Tag'* entry for that row or '0' otherwise. This is, essentially, one-hot encoding.

Test results, however, showed no improvement with this feature included as we have encoded it. We may have seen improvements if we created a feature to associate the '*User Tag'* with some attributes of the website on which the advertisement was presented. However, we could not determine a reasonable way to do this since the features that signified any of the webpage's properties had very large value spaces ('*URL'* and *'Domain',* for example).

The final feature list contains the following:

- *Weekday*
- *User-agent*
- *Ad area*
- *Ad slot visibility*
- *Hour*
- *Ad exchange*
- *Ad slot format*
- *Ad slot floor price*
- *Domain*

To further eliminate features at this point, we had to resort to automated methods since these features, when one-hot encoded resulted in a rather large feature space of roughly 80,000 columns. We used sklearn's **SelectPercentile** utility which runs a univariate statistical analysis on each feature, and transforms the training set to use the top 10% percent scoring features. The statistical test we utilized is the ANOVA f test.

3. Forecasting Model

We are attempting to classify samples into two categories: 'click', 'no click', as such, we resorted to testing classification algorithms and evaluated their performance using cross-fold validation. Since the data-set has widely unbalanced classes, we resorted to using stratified-folds which maintain the same proportion of 'click' and 'no click' across folds; however, we have to be careful in selecting our number of folds since the number of 'click' values in each partition would decrease with increasing number of folds, possibly leading to erroneous results. As a balance of sorts, we decided to use the stratified-folds cross validation with 5 folds where each fold consisted of training on 4 partitions, 80%, of the data and testing on the remaining partition, 20%, of the data.

At this stage and after testing using the cross-fold validation, we have eliminated additional features such as, the created '*Location*' feature which consisted of a concatenation of the '*Region*' and '*City*' features. This elimination occurred since it did not show any improvement in our score.

The algorithms we tested include: Support Vector Machines, Boosted Trees Classifier, and Logistic Regression.

Support Vector Machines [3]

SVM are a set of supervised learning methods that could be used for classification tasks. Support Vector Machine classifier performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1.

Boosted Tree Classifier [1]

The Boosted Trees Model is a type of additive model that makes predictions by combining decisions from a sequence of base models. The boosted trees model is very good at handling tabular data with numerical

features, or categorical features with fewer than hundreds of categories. Unlike linear models, the boosted trees model are able to capture non-linear interaction between the features and the target.

Logistic Regression [2]

Logistic regression is a discriminative probabilistic classification model that operates over real-valued vector inputs. The dimensions of the input vectors being classified are called "features" and there is no restriction against them being correlated. Logistic regression is one of the best probabilistic classifiers, measured in both log loss and first-best classification accuracy across a number of tasks.

4. Results and Evaluation

We evaluated the performance of the three algorithms using the stratified cross-fold validation discussed above. The metric we used was area under curve (AUC) as it provides a good measure of the efficacy of the model, and is the metric used in the competition as well. The following figures show the ROC Curve of the 5 folds for each of the tested models.

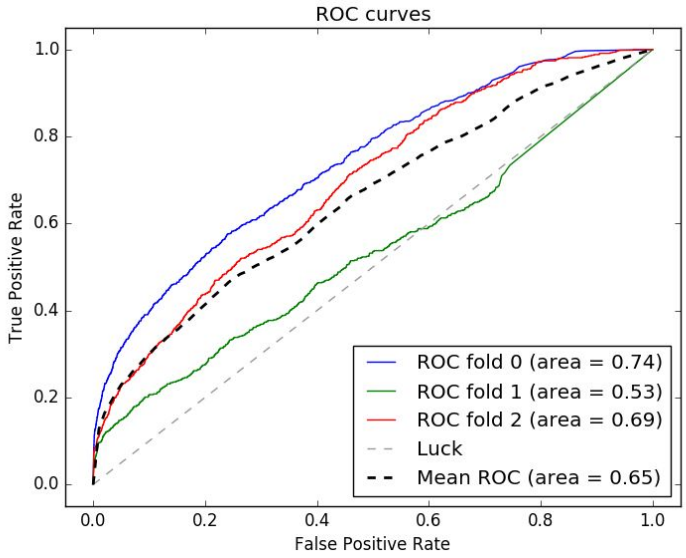

Figure 3: ROC Curves for SVM

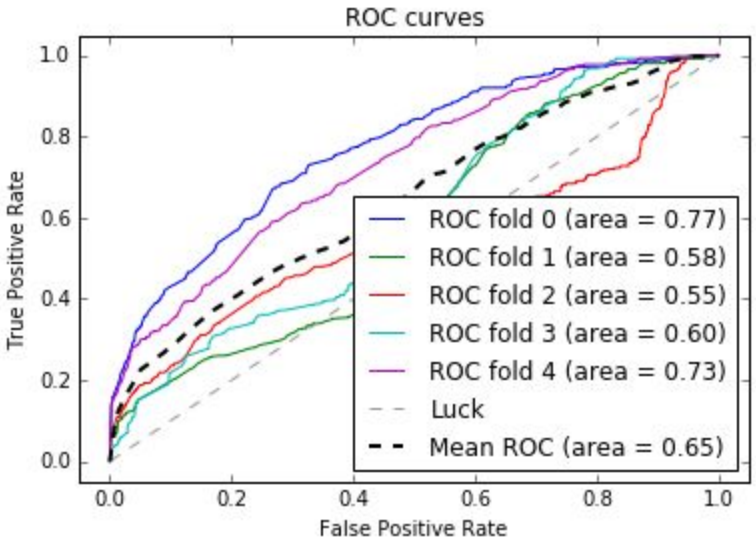

Figure 4: ROC Curves for Boosted Tree Classifier

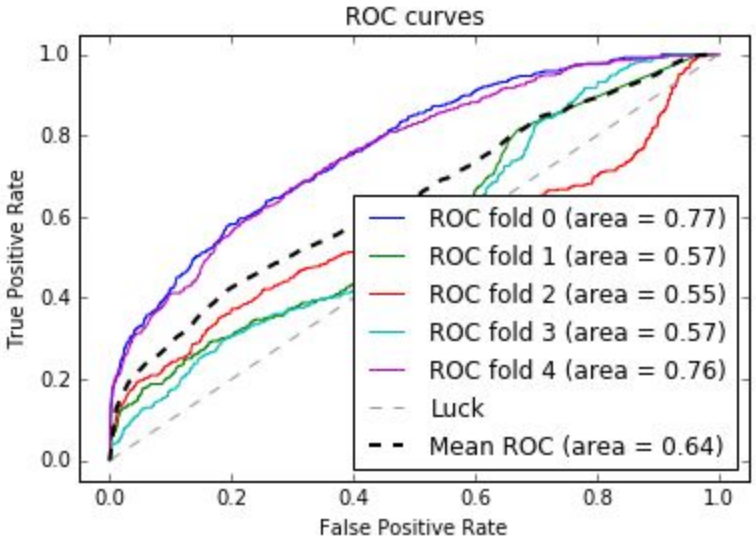

Figure 5: ROC Curves for Logistic Regression

As the figures show, SVM performed worst with an average AUC of 0.56 across folds (we show the 3 folds test here because the 5 folds test resulted in SVM having a mean AUC of 0.5, i.e. SVM performed no better than random guessing). The Boosted Trees Classifier has a slightly higher AUC than the Logistic Regression model; however, when we attempted submissions with each, we achieved a higher score with the Logisitic Regression model. Moreover, the Boosted Trees Classifier took a very long time to train since it is essentially training 200 Logistic Regressions and using them to vote on the prediction. Therefore, we decided to select the logistic regression as the model for the prediction. Our submission using Logistic Regression achieved an AUC of 0.81045 in the competition's public leaderboard which represent approximately 50% of the test data.

Observe that the score achieved is far higher than the average AUC observed during the evaluation. This is a result of the fact that the learner is trained on the entire data-set before creating the submission. Additionally, note in the evaluation above that the Boosted Trees and Logistic Regression Classifiers both performed badly in folds 1, 2, 3 but the results in Folds 0 and 4 are close to 0.81045.

5. Conclusion

Click-through rate estimation plays a vital role for selection of an advertisement. In this challenge, we propose and establish a model to predict the CTRs of advertisements using Logistic Regression. By eliminating features with no predictive value, and performing some simple feature engineering, we were able to achieve 0.81045 in the inclass Kaggle competition's public leaderboard.

References

1. GraphLab Create User Guide https://dato.com/learn/userguide/index.html.
2. Lingpipe: http://alias-i.com/lingpipe/demos/tutorial/logistic-regression/read-me.html.
3. Statsoft: http://www.statsoft.com/Textbook/Support-Vector-Machines.