Predicting Clicks:
Estimating Click-Through Rate for New Ads

Harika Konagala
University of Maryland, Baltimore County
hk12@umbc.edu

ABSTRACT
Search engine advertising has become a significant element of the Web browsing experience. Choosing the right ads for the query and the order in which they are displayed greatly affects the probability that a user will see and click on each ad. This ranking has a strong impact on the revenue the search engine receives from the ads. Further, showing the user an ad that they prefer to click on improves user satisfaction. For these reasons, it is important to be able to accurately estimate the click-through rate of ads in the system. For ads that have been displayed repeatedly, this is empirically measurable, but for new ads, other means must be used. Through this paper, I report the results from click-through rate prediction. To achieve this, I have trained the data with Logistic Regression model by aggregating various features and made different combinations of them. Although the results are just considerable we can understand how each feature helped in learning the model and the reasons behind the way the model functioned.

Keywords: Click-Through Rate, Web advertising, CTR, Session, Ads, Query, CPC, instance, and AUC.

1. INTRODUCTION

Most major search engines today are funded through textual advertising placed next to their search results. The market for these search advertisements (sometimes referred to as "paid search") has exploded in the last decade to $5.75 billion and is expected to double again by 2010 [1]. The most notable example is Google, which earned $1.63 billion in revenue for the third quarter of 2006 from search advertising alone [2] (a brief summary of the history of sponsored search can be found in [3]).

Though there are many forms of online advertising, in this paper I have tried to restrict only to the most common model: pay-per-performance with a cost-per-click (CPC) billing, which means the search engine is paid every time the ad is clicked by a user (other models include cost-per-impression, where advertisers are charged according to the number of times their ad was shown, and cost-per-action, where advertisers are charged only when the ad display leads to some desired action by the user, such as purchasing a product or signing up for a newsletter). Google, Yahoo, and Microsoft all primarily use this model.

To maximize revenue and user satisfaction, pay-per-performance systems must predict the expected user behavior for each displayed advertisement and must maximize the expectation that a user will act (click) on it. The search system can make expected user behavior predictions based on historical click-through performance of the ad. For example, if an ad has been displayed 100 times in the past, and has received 5 clicks, then the system could estimate its click-through rate (CTR) to be 0.05. This estimate, however, has very high variance, and may only reasonably be applied to ads that have been shown many times. This poses a particular problem when a new ad enters the system. A new ad has no historical information, so it is expected click-through rate is completely unknown.

In this paper, the problem of estimating the probability that an ad will be clicked on, for newly created ads is addressed. We can use information about the ad itself (such as the depth, position of the ad and the words it uses), the number of users, and statistics of related ads, to build a model that reasonably predicts the future CTR of that ad.

2. MOTIVATION/ RELATED WORK

The search advertising market has grown significantly in recent years; there are many new advertisers that enter the market each day. Simultaneously, existing advertisers frequently launch new advertising campaigns. Many advertisers create new campaigns each month, some even every day, others create side-by-side orders for testing purposes in order to optimize their ad performance. All of these practices result in an increasing number of ads to be ranked for each query.

Additionally, existing ads are sometimes targeted to new queries. Some advertisers attempt to increase their return on investment by targeting thousands of infrequently searched terms. There has been a significant increase in keyword volume for PPC campaigns: In one study, the number of keywords per campaign per month increased from 9,100 in September 2004 to 14,700 by March of 2005 and was expected to grow to as many as 17,300 by September 2005 [4]. As a result, there is a large inventory of ads for which the search engine has no prior information. These ads need to be ranked with other, already established ads. An incorrect ranking has strong effects on user and advertiser satisfaction as well as on the revenue for the search engine. Thus, for ads that are new, or have not been shown enough times, we must find a way to estimate the CTR through means other than historical observations. This is the goal of the system described in this paper: to predict, for new ads, the probability that an ad will be clicked.

Previous research by Regelson and Fain [5] estimates the CTR of new ads by using the CTRs of existing ads with the same bid terms or topic clusters. From my research online, I came to know that even within the same term there can be a large variation in ad performance (in some cases, the CTR of the best ad can be ten times that of the average ad). To account for these within-keyword variations, it is important to incorporate features that depend on more than just the terms the ad was bid on, the model I used incorporates such features, demonstrated in later sections. The remainder of the paper is as follows. First, we discuss the term CTR. The next two sections describe the dataset and model. Section 6 introduce features to the model, Bag of words concept, Re-sampling techniques, and AUC metric. In Section 7, we discuss the results and note observations about the model performance. And then conclude with a summary of contributions and future work.

3. CLICK-THROUGH RATE

Whenever an ad is displayed on the search results page, it has some chance of being viewed by the user. The farther down the page, an ad is displayed, the less likely it is to be viewed. As a simplification, we consider the probability that an ad is clicked on to be dependent on two factors: a) the probability that it is viewed, and b) the probability that it is clicked on, given that it is viewed.

Thus, CTR is the metric used to measure ad performance, calculated by the number of times an ad is clicked (clicks) divided by the number of times the ad was shown (impressions).

$$\text{CTR} = \frac{clicks}{impressions}$$

This is the value I want to estimate since it provides a simple basis for comparison of competing ads. The goal is to create a model which can predict this CTR for new ads. In the next sections, I first present the data which is used to train and test the model, followed by details on the model itself.

4. DATASET

The data I used is from the KDD cup [6], held by Soso.com. The training data file is a text file, which contains all the information for each instance and it is the main file in this project. To understand the training data, let us begin with a description of search sessions.

A search session refers to an interaction between a user and the search engine. It contains the following ingredients: the user, the query issued by the user, some ads returned by the search engine and thus impressed (displayed) to the user, and zero or more ads that were clicked by the user. For clarity, we introduce a terminology here. The number of ads impressed in a session is known as the 'depth'. The order of an ad in the impression list is known as the 'position' of that ad. An Ad, when impressed, would be displayed as a short text known as 'title', followed by a slightly longer text known as the 'description', and a URL (usually shortened to save screen space) known as 'display URL'.

Each session is divided into multiple instances, where each instance describes an impressed ad under a certain setting (i.e., with certain depth and position values). Then the instances are aggregated with the same user id, ad id, query, and setting in order to reduce the dataset size. Therefore, schematically, each instance contains at least the following information: User id, ad id, query, depth, position, impression, clicks.

Where the impression and clicks are the number of search sessions in which the ad (AdID) was impressed by the user (UserID) who issued the query (Query) and the number of times, among the above impressions, the user (UserID) clicked the ad (AdID) respectively.

Moreover, the training, validation, and testing data contain more information than the above list, because each ad and each user have some additional properties. Some of these properties are included in the training, validation, and the testing instances, and put other properties in separate data files that can be indexed using ids in the instances.

Finally, after including additional features, each training instance is a line consisting of fields delimited by the TAB character:

- **Click:** as described in the above.
- **Impression:** as described in the above.
- **DisplayURL:** a property of the ad.
  - The URL is shown together with the title and description of an ad. It is usually the shortened landing page URL of the ad, but not always. In the data file, this URL is hashed for anonymity.
- **AdID:** as described in the above list.
- **AdvertiserID:** a property of the ad.
  - Some advertisers consistently optimize their ads, so the title and description of their ads are more attractive than those of others' ads.
- **Depth:** a property of the session, as described above.
- **Position:** a property of an ad in a session, as described above.
- **QueryID:** id of the query.
  - This id is a zero-based integer value. It is the key of the data file 'queryid_tokensid.txt'.
- **KeywordID:** a property of ads.
  - This is the key of 'purchasedkeyword_tokensid.txt'.

- **TitleID:** a property of ads.
  - This is the key of 'titleid_tokensid.txt'.
- **DescriptionID:** a property of ads.
  - This is the key of 'descriptionid_tokensid.txt'.
- **UserID**
  - This is the key of 'userid_profile.txt'. When we cannot identify the user, this field has a special value of 0.

4.1 ADDITIONAL DATA FILES

There are five additional data files, as mentioned in the above section:

1. queryid_tokensid.txt
2. purchasedkeywordid_tokensid.txt
3. titleid_tokensid.txt
4. descriptionid_tokensid.txt
5. userid_profile.txt

Each line of the first four files maps an id to a list of tokens, corresponding to the query, keyword, ad title, and ad description, respectively. In each line, a TAB character separates the id and the token set. A token can basically be a word in a natural language. For anonymity, each token is represented by its hash value. Tokens are delimited by the character '|'.

Each line of 'userid_profile.txt' is composed of UserID, Gender, and Age, delimited by the TAB character. Note that not every UserID in the training and the testing set will be present in 'userid_profile.txt'. Each field is described below:

1. **Gender:** '1' for male, '2' for female, and '0' for unknown.
2. **Age:** '1' for (0, 12], '2' for (12, 18], '3' for (18, 24], '4' for (24, 30], '5' for (30, 40], and '6' for greater than 40.

Table 1: size of the dataset

File	Size	Records
Training	9.9GB	149,639,105
userid_profile.txt	283 MB	23,669,283
queryid_tokensid.txt	704 MB	26,243,606
Titleid_tokensid.txt	171 MB	4,051,441
Descriptionid_tokensid.txt	268 MB	3,171,830
purchasedkeywordid_tokensid.txt	26MB	1,249,785
Test	1.3GB	20,297,594

4.2 TESTING DATASET

The testing dataset shares the same format as the training dataset, except for the counts of ad impressions and ad clicks that are needed for computing the empirical CTR.

5. MODEL

Since the goal is to predict a real-value (the CTR of an ad), we can treat it as a regression problem – that is, to predict the CTR given a set of features. I chose to use logistic regression, which is ideally suited for probabilities as it always predicts a value between 0 and 1:

$$\text{CTR} = \frac{1}{1+e^{-z}} \quad Z = \sum_i wi\, fi(ad)$$

Where *fi(ad)* is the value of the *i*th feature for the ad, and *wi* is the learned weight for that feature. Features may be anything, such as the number of words in the title, the existence of a word, etc. (They will be described in more detail in the next sections.)

The evaluation metric applied to measure the model's performance is the Area Under Curve (AUC), which is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. The implicit goal of AUC is to deal with situations where you have a very skewed sample distribution, and don't want to overfit to a single class.

AUC is not always area under the curve of a ROC (Receiver Operating Characteristic) curve. In the situation where you have imbalanced classes, it is often more useful to report AUC for a precision-recall curve.

In preliminary experiments, I measured the performance using decision trees. They were found to have no significant improvement over logistic regression. Thus, for ease of interpretation and simplicity, I continued with logistic regression for the remainder of the experiments (I present only the logistic regression results here). In the next section, we will discuss the first set of features, intended to capture the CTR variance.

6. APPROACH

6.1 UNDERSTANDING THE DATA

Initially, I started off with data munging followed by feature extraction and feature selection. In data munging, I tried to learn the structure of the data first. How are the values distributed among all the columns. Are there any hidden patterns. Are there any missing values and duplicate records. How can I make use of these columns as input features to the model. All these sorts of questions were answered when I explored the data.

In machine learning, Feature extraction is a technique that creates new features from functions of the original features, whereas feature selection returns a subset of the features. Following the definition given by Wikipedia, I used the categorical features advertiserid, adid and userid in the model by aggregating users and ads for each unique advertiserid. Further, in feature selection process, I tried different combinations of features. Firstly merged the gender, age columns from userid_profile table and then, as the dataset had four tables with anonymized hash values, I implemented the bag of words model in order to make use of them.

6.2 BAG OF WORDS

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

In our case, we are maintaining a dictionary of hash values with their frequency of occurrence. To better understand, an example record from the titleid_tokensid table [2]: For the titleid 12, the corresponding pipe separated tokens are 234, 65, and 530. In the bag of words model the frequency of occurrence of each token is calculated by parsing the entire title table, shown in table [2]. In the next step, all the frequencies of tokens in a record are added up, as seen in table [3]. Thus, every titleid is associated with the token score calculated from the hash value dictionary further replacing the tokens

column, as seen in table [3]. The same process is repeated for all other tables with hash values. But the noticeable problem with this model is that there are few tokens which are most frequently repeated, i.e., their occurrence may not have a significant effect on the CTR prediction. Therefore, such tokens can be removed by setting a threshold value. If the threshold is set to 90 then the top 10% tokens are counted as zeros. By varying the threshold value we can tune the input features to the model. Like in the above example, token 65 has the highest frequency of 906582. In text format this token could be any word like 'the', 'an' etc. such words can always be ignored and still make no difference which is what I did.

Table 2: Example

Titleid	Token score
12	1167809

Table 3: Frequency table

Tokens	Frequency
234	257649
65	906582
530	3578

Table 4: Transformed table

Titleid	Tokens
12	234|65|530

6.3 CORRELATIONS

After looking at the data, it is evident that there are some significant patterns such as in Figure 1 we can see which features received the highest weight, and which have the lowest weight (or highest negative weight). Features like depth, position, gender, age have higher weight followed by aggregated features and then the hash values in CTR prediction. Whereas when we look at the factor plot in Figure 2 we can observe the age and gender distribution based on the click rate. There are many interesting insights from this particular plot. Some of them are men from age 12 to 30 are active web browsers and watch more ads compared to other ages. Similarly, women from age 12 to 40 years are also super active web browsers. From both the graphs, we can conclude that women watch more ads compared to men and had consistently high click rate as well.

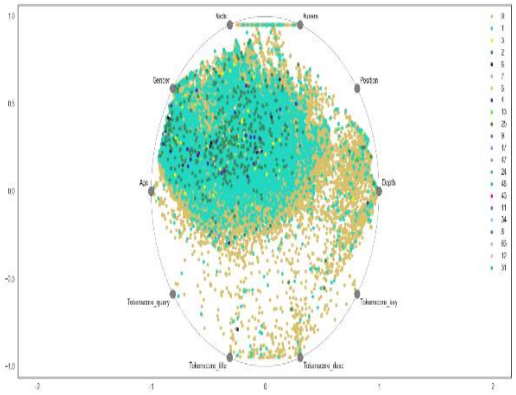

Figure 1. Click rate distribution w.r.t all features

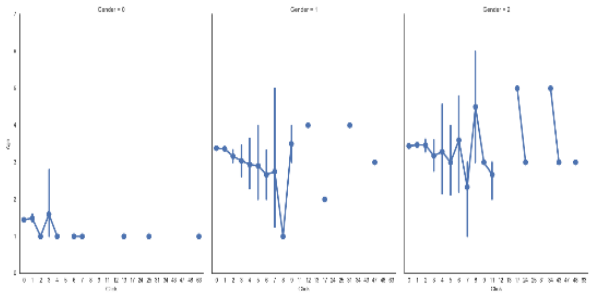

Figure 2. Click rate distribution w.r.t age, gender

Figures 3 and Figure 4 show that comparatively there is a higher chance of clicking the ad when they are placed at a depth and position of 2 respectively. Apparently, the statistics show that the number 2 has better placement and is most convenient for clicking.

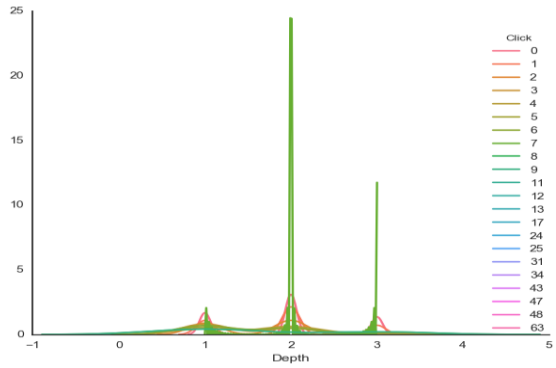

Figure 3. Depth of an ad Vs click rate

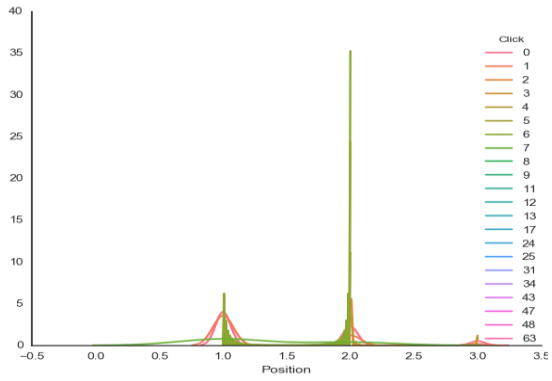

Figure 4. Position of an ad Vs click rate

6.4 RE-SAMPLING

So far so good but through this project, I'm trying to tackle a multi-class problem. Since I have to predict CTR, the number of clicks could be any number, not just 0 and 1 (means not-clicked and clicked). And Most of the classification datasets do not have an exactly equal number of instances in each class, the same happened with this data as well. Thus, leading us to a term called 'Imbalanced data' [7]. Imbalanced data typically refers to an issue in classification problem where the classes are not represented equally.
For example, if we look at our dataset as a 2-class (binary) classification problem with 100 instances (rows). A total of 94 instances are labeled with non-clicked and the remaining 6 instances are labeled with clicked. This is an imbalanced dataset and the ratio of non-clicked to clicked instances is 94:6 or more concisely 15:1. Although you can have a class imbalance problem on two-class classification problems as well as multi-class classification problems. There are a number of techniques to tackle this issue like cost-sensitive algorithms, SMOTE analysis, Random oversampling and undersampling and a combination of over and under sampling can also be done. For this data, I have tried applying random oversampling and undersampling, SMOTE analysis.

Oversampling and undersampling in data analysis are techniques used to adjust the class distribution of a dataset. Oversampling and undersampling are opposite and roughly equivalent techniques. They both involve using a bias to select more samples from one class than from another. The usual reason for oversampling is to correct for a bias in the original dataset. For example, suppose we have taken the first 1000 rows from the data of which 94% are non-clicks and the rest 6% are distributed with other click values. We know that the class is biased, and we may wish to adjust our dataset to represent this.

Simple oversampling will select each clicked example multiple times, and this copying will produce a balanced dataset of equal samples. Simple undersampling will drop some of the non-clicked samples at random to give a balanced dataset of equal samples. In this process, the total sample size of 1000 records may vary according to the number of clicked and non-clicked samples.

SMOTE (Synthetic Minority Over-sampling Technique), as its name suggests, SMOTE [7] is an oversampling method. It works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances.

6.5 AUROC METRIC

Once the data is ready and the model is built we need a metric to measure the performance of the model. As explained before in section 5, I have chosen AUROC (Area Under Receiving Operating Characteristic Curve) [12] because it works well with Imbalanced data.

Two things very important in this metric are:

- True positive rate (**TPR**), aka. Sensitivity, hit rate, and recall, which is defined as the metric that corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points. In other words, the higher TPR, the fewer positive data points we will miss.

$$\text{TPR} = \frac{TP}{TP+FN}$$

- False positive rate (**FPR**), aka. Fall-out, which is defined as the metric that corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points. In other words, the higher FPR, the more negative data points we will miss-classified.

$$\text{FPR} = \frac{FP}{FP+TN}$$

In the case of multi-class, we calculate TPR and FPR for each classifier and plot them on one plot.
Now once the measure is evaluated, how can we interpret the performance of AUC. For that consider a graph with a total area of 100%

- AUC = 1 – perfect classifier for which all positive come after all negatives
- AUC = 0.5 – randomly ordered
- AUC = 0 – all negative come before all positive

So AUC ∈ [0, 1]

Interpretation: Formally, the AUC of a classifier C is the probability that C ranks a randomly drawn "+" example higher than a randomly drawn "−". Example. I.e. AUC(C) = $P[C(x^+) > C(x^-)]$

To sum up the approach:

- Firstly, cleaned the data and then extracted useful features from the training table
- Used bag of words model to use the hash values as input features
- One hot encoding for categorical features like depth, position, age, gender
- Merged different combinations of tables with and without threshold limit on different data sizes such as 10000, 50000, 100000, 500000, 1400000 records
- Applied sampling techniques on all combinations

- Tabulated the performance for each run in terms of AUC score

7. RESULTS AND DISCUSSION

7.1 OBSERVATIONS

After running the logistic regression model inclusive of all the features, my initial results in predicting clicks and impressions were 94.6 and 89 respectively which is apparently a very good score but when I checked the predicted values, I realized that the model was biased. That's when I applied the random sampling [8]. And then my results dropped down to 45 and 26.7 for oversampling, 43 and 24 for undersampling respectively.

Hence, I discovered that the accuracy metric I used was not appropriate to measure the performance of imbalanced dataset. The results are summarized in Figure 5.

Then I shifted to AUC score and ran a logistic regression model on a variety of features varying the size of data. My first observation was: I achieved an AUC score of 50 without sampling on all the combinations I tried which is no better than random guessing.

After applying the random sampling:

- Case 1: Merged user table with training file. In this case, the oversampling technique showed better performance compared to undersampling. And one noticeable observation is the AUC score is linearly incrementing with the size of the data. The results are summarized in Figure 6.
    - Reasons for low score: as we haven't included all the features, in this case, it could be one reason for a low score. Second being the size of the data. Since the model hasn't been trained on the entire dataset, it gave average results.
- Case 2: Used the tables containing hash values with no threshold limit along with the training file. In this case, the undersampling technique showed better results. Even here, the performance is proportional to the data size. The reasons for low score are same as in case 1. The results are tabulated in Figure 7.
- Case 3: Built the model on all the features. The performance, in this case, is also linearly incremental w.r.t the data size and this model worked well with a highest AUC score of 63 using oversampling and 56 using undersampling. The results are summarized in Figure 8.
- Some of the common observations are: In all the cases the model was able to predict very good CTR values but the scores were little low. This is mainly due to the size of the data. Looking at the linearity in the results, I can be sure that if I trained the model on the entire dataset I would achieve a better score.
- SMOTE Analysis:
    - Apart from random sampling, I also tried SMOTE [8] on case 3 which gave a score of 51 again no better than guessing. The reason for such a low performance by SMOTE is: it is usually better with entire data because it uses kmeans algorithm which requires each click value to have samples at least equal to the total number of unique click values in the data.

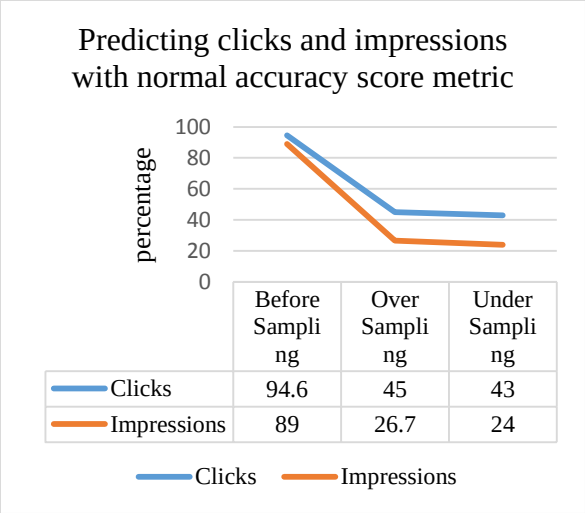

Figure 5. Initial Results for Clicks and Impressions

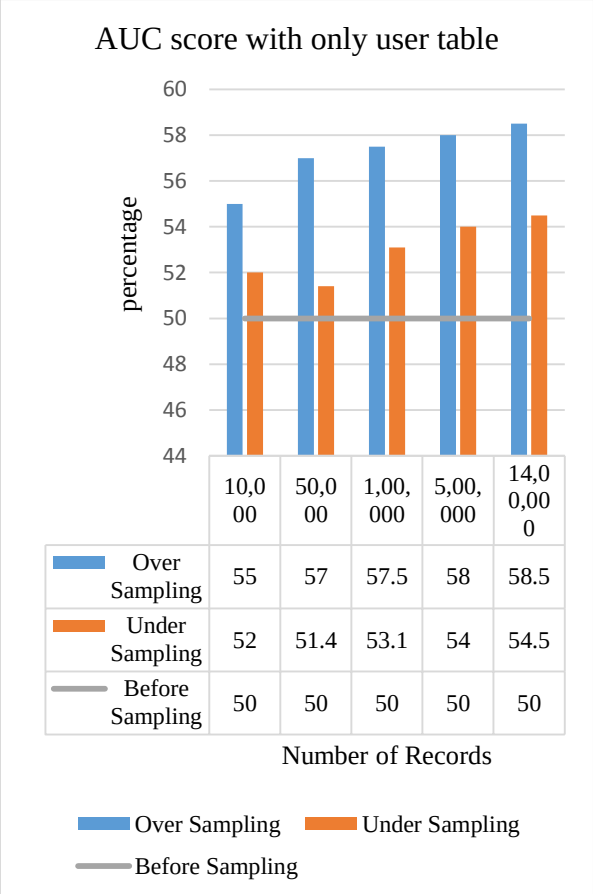

Figure 6. AUC score with only user table merged with training file

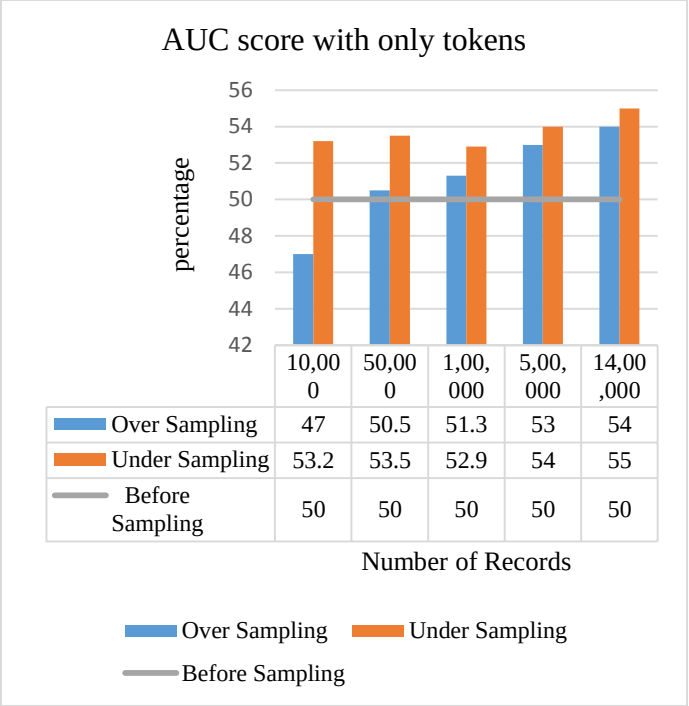

Figure 7. AUC score with query, title, description, keyword tables with no threshold limit merged with training file

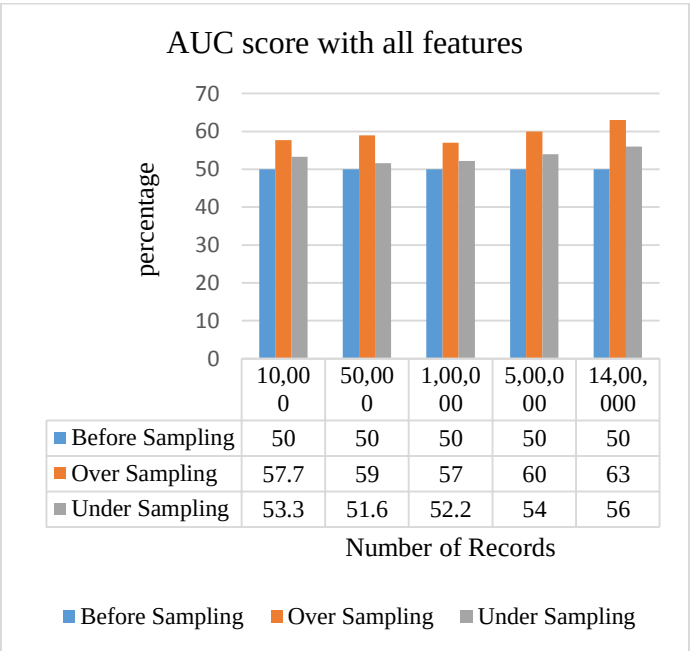

Figure 8. AUC score with all the features

7.3 ROAD BLOCKS

Building the model on the entire dataset was the biggest road block I have faced in this project. The data was very huge with ~140 million records. Due to memory constraints, I was not able to process the entire data on my machine. Although I tried loading the data onto BlueWave cluster each iteration was taking too long to finish, as we know that dealing with real world data can be a tedious task sometimes. I have decided to work with just $1/100^{th}$ of the data.

8. FUTURE WORK

Many different tests and experiments have been left for the future due to lack of time (i.e. the experiments with real data are usually very time-consuming, requiring even days to finish a single run). There are some ideas that I would have liked to try during the entire process. This project so far has been mainly focused on building a logistic regression model to predict CTR and measure the performance by considering a variety of input features. Although the results were not very good, it could be more interesting to try the following:

1. To work on the entire dataset and see if there is any boost in the performance.
    a. Due to memory problems, I have trained only on $1/100^{th}$ of the entire dataset.
2. Try other algorithms like SVM, KNN or maybe a combination of Decision Trees and Logistic Regression to train my model and compare the results.
3. I have completely ignored the DisplayURL feature in my model, if the dataset could be little more elaborate then including features like how often users have visited the ad's display URL and its domain, how long they remain on that page, whether they click "back" or a link off of the page, etc. could prove useful.
4. Another source of information could be human judges. I would like to see if a brief (5 second) "instinctive" judgment by a person could be a useful feature to the model. Since the decision to click or not is based on even less than this amount of time on behalf of the end user, I belief such quick human judgments could provide significant value to the model while incurring a low over-head cost.

9. CONCLUSIONS

A good initial estimate of an advertisement's click-through rate is important for an effective online advertising system. I have presented a logistic regression model that achieved a 63% of AUC score with oversampling and 56% with undersampling. Although the results are not very good the model is easy to understand, quick to train, and efficient enough to be tested by anyone interested. Besides the performance of the model personally, the entire project has been a huge learning experience for me where I came across many unknown concepts like SMOTE, Imbalanced data, AUC metric and much more.

10. REFERENCES

1. D. Murrow, "Paid Search Ad Spend Will Hit $10 Billion By 2009" In eMarketer, http://www.emarketer.com/Article-.aspx?1003861.

2. L. Baker, "Google vs. Yahoo: Earnings Reports Comparison," In Search Engine Journal http://www.searchengine-journal.com/?p=3923.

3. D. Fain and J. Pedersen. "Sponsored Search: a Brief History", In Proceedings of the Second Workshop on Sponsored Search Auctions, 2006.

4. E. Burns, "SEMs Sees Optimization PPC", In ClickZ,

http://www.clickz.com/showPage.html?page=3550881.

5. M. Regelson and D. Fain, “Predicting click-through rate using keyword clusters,” In Proceedings of the Second Workshop on Sponsored Search Auctions, 2006.

6. Tencent and K. Cup, "KDD cup 2012, track 2,". [Online]. Available: http://www.kddcup2012.org/c/kddcup2012-track2. Accessed: Dec. 22, 2016.

7. J. Brownlee, "8 tactics to combat Imbalanced classes in your machine learning Dataset," in *Machine Learning Process*, Machine Learning Mastery, 2015. [Online]. Available: http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/. Accessed: Dec. 22, 2016.

8. imbalanced-learn, "Welcome to imbalanced-learn documentation! — imbalanced-learn 0.2.0.dev0 documentation," 2016. [Online]. Available: http://contrib.scikit-learn.org/imbalanced-learn/. Accessed: Dec. 22, 2016.

9. myui, "Myui/hivemall," GitHub, 2016. [Online]. Available: https://github.com/myui/hivemall/wiki/KDDCup-2012-track-2-CTR-prediction-dataset. Accessed: Dec. 22, 2016.

10. coef _, "CostSensitiveLogisticRegression — costcla documentation," 2016. [Online]. Available: http://albahnsen.com/CostSensitiveClassification/CostSensitiveLogisticRegression.html. Accessed: Dec. 22, 2016.

11. 2016, "Documentation scikit-learn: Machine learning in python — scikit-learn 0.18.1 documentation," 2010. [Online]. Available: http://scikit-learn.org/stable/documentation.html. Accessed: Dec. 22, 2016.

12. "ROC analysis," 2015. [Online]. Available: http://mlwiki.org/index.php/ROC_Analysis#AUC:_Area_Under_ROC_Curve. Accessed: Dec. 22, 2016.

13. Matthew Richardson, Ewa, Robert Ragno, “Predicting CTR”, In Proceedings of the Second Workshop on Sponsored Search Auctions, 2006.

14. Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi ,Antoine Atallah, Ralf Herbrich, Stuart Bowers, Joaquin Quiñonero Candela, “Practical Lessons from Predicting Clicks on Ads at Facebook”

15. Weinan Zhang, Shuai Yuan, Jun Wang, Xuehua Shen, “Real-Time Bidding Benchmarking with iPinYou Dataset”