Northeastern University

CLICK-THROUGH RATE PREDICTION OF ADS

Shubhi Singal - singal.shu@husky.neu.edu
Manpreet Kaur - lnu.manp@husky.neu.edu

December 12, 2019

Class: ALY6020- **Predictive Analytics

Professor: **Marco Montes de Oca

Summary

We used the train data from the Kaggle competition to predict the probability, that given some key information about the ad presented to a user on the phone, he will click on the Ad. Since Ads are major revenue generators of major Internet companies, being able to predict the probability of click is an important study. Revenue is the product of the bid and probability of the ad getting clicked. Out of the two variables, bid is what the company decides, leaving us with controlling the revenue based on the ability to predict the click. This is a machine learning problem is a classification problem with probabilistic outputs.

Our machine learning objective which determined how good our model was log- loss. We log-loss instead of accuracy or Area under the Curve is because log-loss penalises, misclassifications. Also, since revenue is the multiplication of bid and probability of click, log-loss was finalized as the best approach. Where, AUC could have been used, if the problem statement was to determine the order of the ad placement.

Positional bias is which the position the Ad is placed could impact the model, is what we had assumed. So, we decided to add weights which were inversely proportional to the historical CTR. However, this only reduced the performance of the model, although we had assumed it should have improved.

We focused on finding a model which would fit in the machine learning constraints of an Ad click type problem, which are, low latency, relevance, interpretability, parallelizability. We understood various models before implementing logistic regression and gradient boosting model. Naive Bayes model, although simple to run, low latency and high interpretability, assumes that the features are gives probability value which are vague. KNN model has a very high run time complexity and hence, ignored it. Linear support-vector machines models are not optimizing explicitly for log-loss and Kernel SVM were very hard to train. We also considered stacking model, but since low latency was one of our criteria, we decided to not go ahead with that. Logistic Regression was a major choice since it has low latency and most importantly, the model internally tries to minimize the log-loss value, However, the drawback was the it was highly likely that the data is not linearly separable, and such was the case with our dataset. We went ahead and implemented the Gradient Boosting algorithm, which after training gave us a log loss value of 0. 40.

Introduction

In today's world we all make a minimum of 2 searches each day and there are a few adds each time we make a search. It is interesting to note that 8% of the times the first Ad gets clicked, 5% times the second Ad gets clicked and 3% times the third Ad gets clicked. If the right Ad is shown to the user, there are high chances he/she will click on the Ad. The probability of clicks varies depending on country, medium used (laptop or phone) and time of the day. The dataset has limited our discussion to predicting the click-through rates of Ads on the phone only.

Ads being clicked impact the multi dollar business of many vast companies. Online giants like Google have a major share of their revenue coming from Ad being clicked. Probability of Ad click is a very important metric for evaluating Ad performance. This is a very interesting study and helped us gain insights into lot of facts of google search the thought process of building the right model for the data.

The data was collected from - https://www.kaggle.com/c/avazu-ctr-prediction/data

We tried to understand, even before applying the machine learning algorithm of models upon the data, which models would be appropriate on the dataset and the given business problem.

Since the train data itself was very large- 10 GB, we used the train dataset itself for both training and testing by splitting it in appropriate ratios. From a 10 million rows large dataset, we randomly sampled it to 2 million, for our analysis. This data was split into train and test to give results for applying machine learning models.

The data was highly unbalanced on the Target class. We used Near miss for under sampling and SMOTE for oversampling and applied logistic regression on both. However, the log-loss value was higher on the unbalanced dataset. So, we went ahead and applied Gradient boosting on the unbalanced dataset to give us a much lower log loss value, close to 0.

Implementation

- Mapping Business problem to Machine Learning problem:

1. What type of machine learning problem is it?

   There are majorly 5 types of ML problems. Reinforcement learning, classification, regression, clustering and recommendation. This is a classification type of problem with probabilistic outputs.

2. What is the machine learning objective?

   The objective is to minimize the log loss of the output. Another fancy name for log loss is binary cross entropy. We might need to use calibrated model on top of base model to correct the probability. We use log loss instead of AUC as it penalizes misclassification. In the case below, where Y_i is the expected value and Y_i1 and Y_i2 are the probabilistic values of the 2 model, the AUC value is 1 for both the cases, whereas Log loss has Val_1 and Val_2, where Val_1 < Val_2.

	Y_i	Y_i1	Y_i2
	1	0.99	0.9
	0	0.12	0.4
	1	0.81	0.6

AUC		1	1
Log- Loss		**Val_1**	**Val_2**

   Since revenue of the company from the Ad is the product of probability of Ad click and bid, log-loss value gives better evaluation metric of the model (since we are multiplying the two variables). AUC metric would be convenient is we were interested in purely the order of the Ad placement instead of revenue generated.

3. Which were our machine learning constraints?

   A Low latency - User would not want to wait till for the algorithm to run to churn the

best add and delay the search result or App to load in.

B Relevant - The better the prediction higher would be the relevance of the result ot the Ad.

C Interpretability **-** It is the degree to which a human can comprehend the cause of decisions and predictions made.

D Parallelizable - Multiple Ads needs to be shown at the same time to different users. Same user on a different App or search may need a different Ad. Hence strong parallelizability is needed to achieve this. Parallelizability during training is a tougher concept to delve in compared to during evaluation.

- Data Description:

We can group all the features in the data into the following categories:

- **Target feature**: click
- **site features**: site_id, site_domain, site_category
- **app feature:** app_id, app_domain, app_category
- **device feature**: device_id, device_ip, device_model, device_type, device_conn_type
- **anonymized categorical features**: C1, C14-C21
- **Time feature**: year, month, date, day, time (YYMMDDHH, so 14091123 means 23:00 on Sept. 11, 2014 UTC)
- **Ad location**: banner_pos

The data was very large (10GB). We either had to use Big Data tools to be able to analyze and make predictions, or randomly sampled to be able to use in the local machine. We chose the 2nd method to go ahead with our problem.

Our data was clean, and no changes were needed in that end. We sampled it down to 2million rows. On a closer look, it was an unbalanced data, so we used Near Miss and SMOTE to balance it out. However, we carried out the Data analysis, before the data balancing.

- Feature Engineering:

Hash Trick:

Most of our data was high cardinality categorical variables. Directly converting it to integer values was an inefficient method, as it uses those values as weights affecting the model. We used Hash trick method which tunes arbitrary features into sparse binary vector. Even a slight change in categorical value will produce a new hashed output. The concept of Hash collision is to reduce the cardinality of the data. If the number of objects is very large, two objects may have the same hash function. A good hash function will reduce the number of collisions.

Data Balancing:

SInce our first model selected was Logistic regression, which has a bias towards majority class, we went ahead and decided to balance the data. The obvious option seemed to under sample the data, since it was 2million rows large. However, to check the performance of model, through our metric selected, which was log-loss, we decided to under sample and oversample it both. Oversampling has a tendency to overfit and under sampling a tendency to reduce performance due to loss of information. SMOTE however, it deleted less useful majority class. Also, sampling is improving model, especially if the data is highly skewed. However, in our case this could not be considered a reason.

Cyclic Variable:

We had to let out model know that features like days and hours are cyclic in nature. To achieve this, we resolved the values into their sine and cos values. This lets the model know that 0 is close to 24 in case of hour variable.

Model Selection:

We understood various models to check which would fulfil all the machine learning constraints. List of models used for unbalanced data and the reasons for not selecting them.

Naive Bayes model - they are simple to run, low latency and high interpretability, assumes that the features are gives probability value which are vague.

KNN model - it has a very high run time complexity and hence, ignored it.

Linear support-vector machines models - they are not optimizing explicitly for log-loss **Kernel SVM -** they were very hard to train.

Stacking model - they have very low latency although it gives log log loss values

Logistic Regression was a major choice since it has low latency and most importantly, the **model internally tries to minimize the log-loss value**.
However, the drawback was the it was highly likely that the data is not linearly separable, and such was the case with our dataset. So, we went ahead and implemented the **Gradient Boosting algorithm**, which after training gave us a log loss value of 0. 40.

We applied logistic regression on Unbalanced Data, SMOTE balanced data (oversampled) and Near miss balanced data (under sampled). Below are the results of the models formed.

Logistic Regression applied on Unbalanced Data

	precision	recall	f1-score	support
0	0.83	0.99	0.90	498103
1	0.10	0.00	0.01	101897
avg / total	0.71	0.83	0.75	600000

6.041412196781143

Log-loss value = 6.04
Recall = 0

Logistic Regression applied on Oversampled Data by SMOTE algorithm

```
Before OverSampling, counts of label '1': 238271
Before OverSampling, counts of label '0': 1161729

After OverSampling, the shape of train_X: (2323458, 26)
After OverSampling, the shape of train_y: (2323458,)

After OverSampling, counts of label '1': 1161729
After OverSampling, counts of label '0': 1161729
             precision    recall  f1-score   support

          0       0.89      0.46      0.61    498103
          1       0.21      0.72      0.33    101897

avg / total       0.78      0.51      0.56    600000

17.074083263989394
```

Log-loss value = 17.07
Recall = 0.72

Logistic Regression applied on Under sampled Data by Near Miss algorithm

```
Before Undersampling, counts of label '1': 238271
Before Undersampling, counts of label '0': 1161729

/Users/manu/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.c
ore.umath_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d

After Undersampling, the shape of train_X: (476542, 26)
After Undersampling, the shape of train_y: (476542,)

After Undersampling, counts of label '1': 238271
After Undersampling, counts of label '0': 238271
             precision    recall  f1-score   support

          0       0.83      0.32      0.46    498103
          1       0.17      0.69      0.28    101897

avg / total       0.72      0.38      0.43    600000

21.31116422221723
```

Log-loss value = 21.31
Recall = 0.17

Under sampled model output is a bad fit, as log loss value is very high. Comparatively, model formed on oversampled data has a lower log loss value. Surprisingly, on unbalanced data, the log loss value was the better but still does not qualify for a good model.

This could possibly mean that out data was not linearly separable by logistic regression.

We went ahead and implemented Gradient Boosting algorithm. The results after training the model with the right parameters are below

```
[250]   train-logloss:0.40057   eval-logloss:0.401655
[251]   train-logloss:0.400519  eval-logloss:0.401611
[252]   train-logloss:0.400494  eval-logloss:0.401595
[253]   train-logloss:0.400452  eval-logloss:0.401569
[254]   train-logloss:0.400431  eval-logloss:0.401554
[255]   train-logloss:0.400372  eval-logloss:0.401505
[256]   train-logloss:0.400336  eval-logloss:0.401473
[257]   train-logloss:0.400327  eval-logloss:0.401469
[258]   train-logloss:0.400293  eval-logloss:0.401446
[259]   train-logloss:0.400282  eval-logloss:0.401444
```

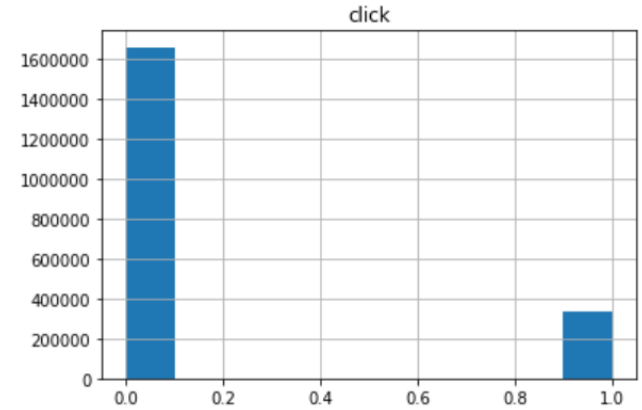
Since the basis of measuring the model performance was log-loss, the gradient boosting model was the right fit for our data, as it gave a low log-loss value = 0.40

Data Analysis

Exploratory data analysis is a way to better understand the data check for abnormalities, identify the missing values, find bias, check the distribution of the data. It is the first and the most important step in any data analysis project.

Our dataset had many categorical features and they were all atomized, thus performing exploratory data analysis seemed like a great choice to better understand the structure of the data. Since the target variable had only 0 or 1 as its value, we counted their frequencies and then plotted a histogram for the same. Counting the values tells us if our data is unbalanced or not. This is the first step to deciding which technique we should use to proceed further.

```
0    0.830503
1    0.169496
Name: click, dtype: float64
```



It is clear from the histogram and the value counts that the dataset is unbalanced. Approximately in the ratio 5:1, meaning that for every person who clicks on the ad there are 5 people who not. Since our data is unbalanced, we now need to decide which techniques to use to help minimize its effect.

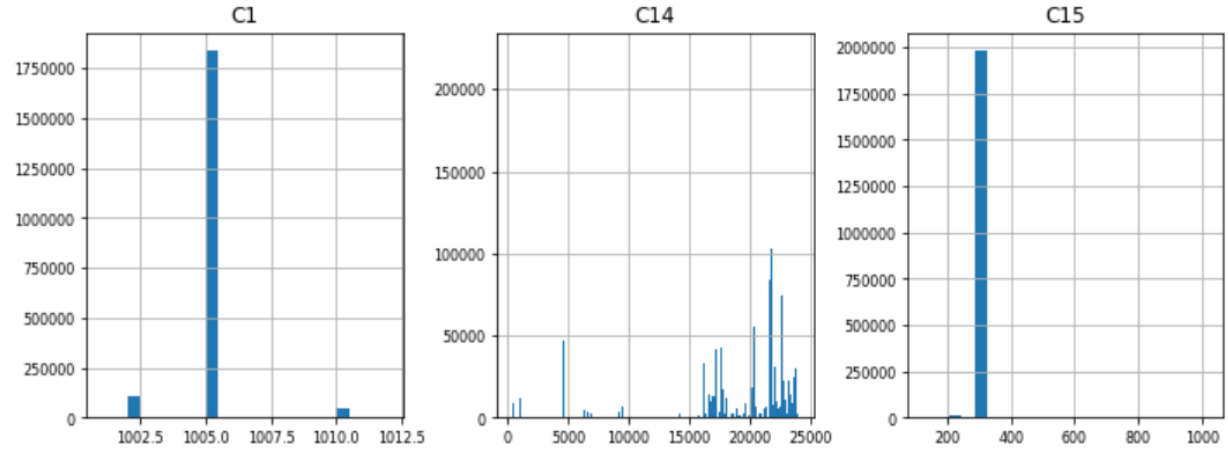
Time is one of the factors which plays an important role on the possibility of an ad being clicked. We were given the data from 10/21/2014 to 10/30/2014, it consisted of the hour, date, month and year.

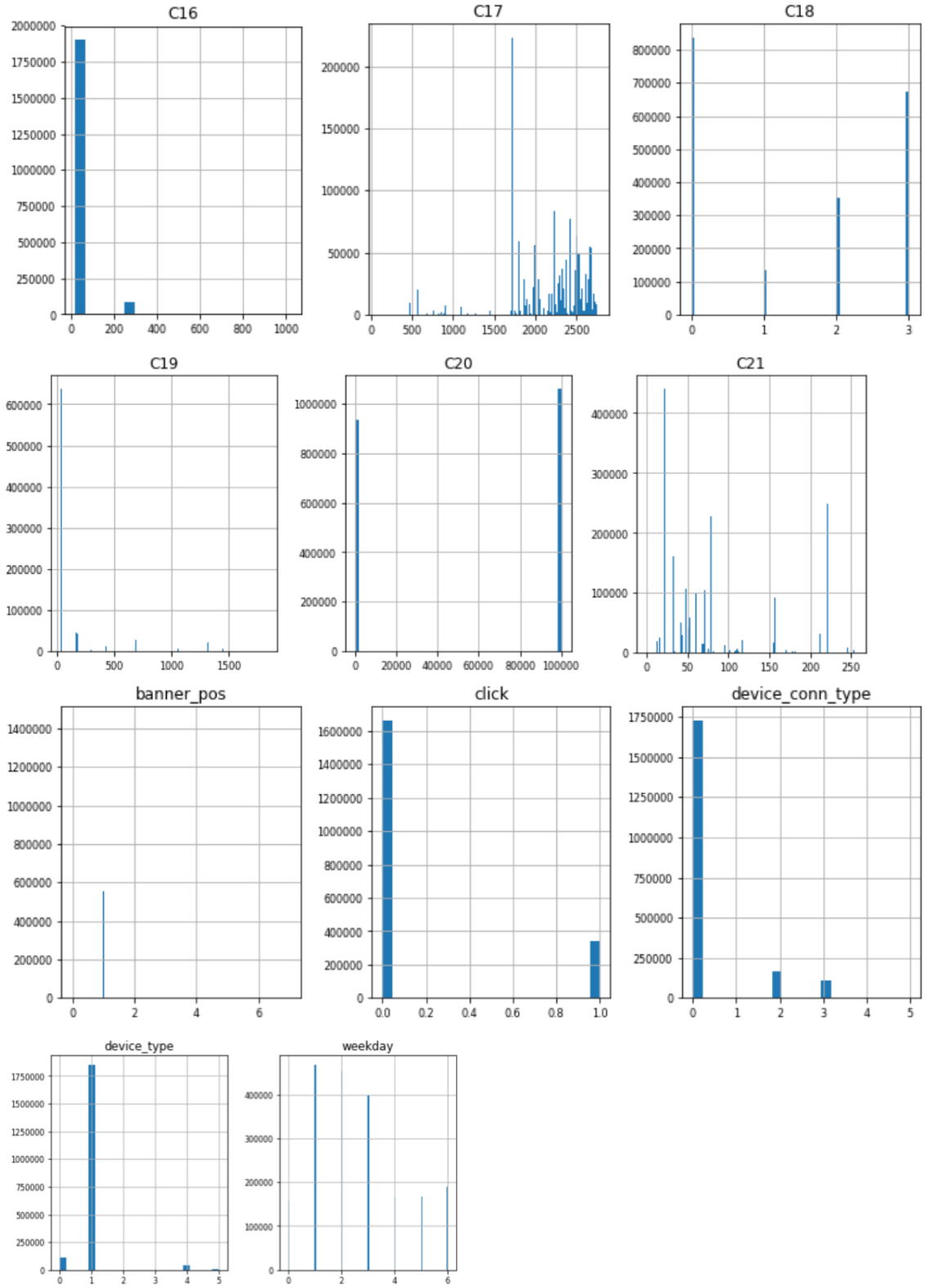
```
count                 2000000
unique                    240
top       2014-10-22 09:00:00
freq                    22244
first     2014-10-21 00:00:00
last      2014-10-30 23:00:00
Name: hour, dtype: object
```

This descriptive analysis of the time column tells us that we have 240 unique values along with the first and the last value that we have.
Since the time column was in the format 'YYYY-MM-DD HH:MM: SS' there was a need to segment it out to different columns.

```
#SPlitting the hour column into usable columns
train_1['weekday'] = train_1['hour'].dt.dayofweek
train_1[['Year','Month','Day']] = train_1.hour.astype(str).str.split("-",expand=True)
train_1[['Day','time']] = train_1.Day.astype(str).str.split(expand=True)
train_1[['hours','minutes','seconds']] = train_1.time.astype(str).str.split(":",expand=True)
train_1 = train_1.drop(['Year', 'Month', 'hour', 'time', 'minutes','seconds'], axis=1)
```

The 'hour' column was split into year, month, day, weekday, seconds, minutes and hours out of these year and month remained the same hence they were dropped. Minutes and seconds columns had no value hence they were dropped too along with the hour column which was the initial column in which the data was stored and finally the time column which was used as an intermediate column to store time.
There are features from C1 to C21 which consists of float, integer or categorical values, and since these variables were anonymized there is very little information on them. To get a better insight into at least some of these values, we separate all the numerical values and plot the histogram for each one of them.

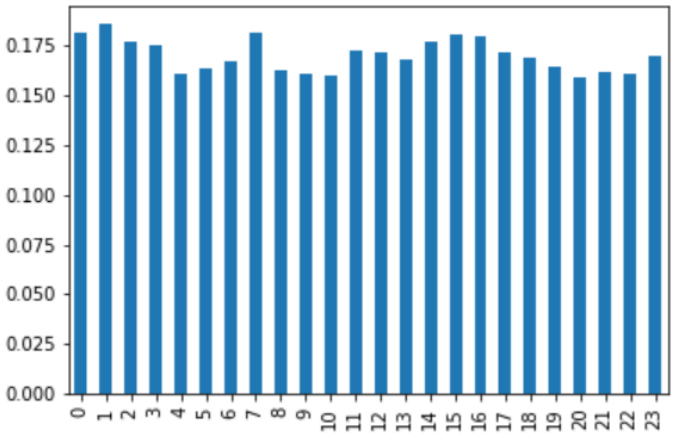




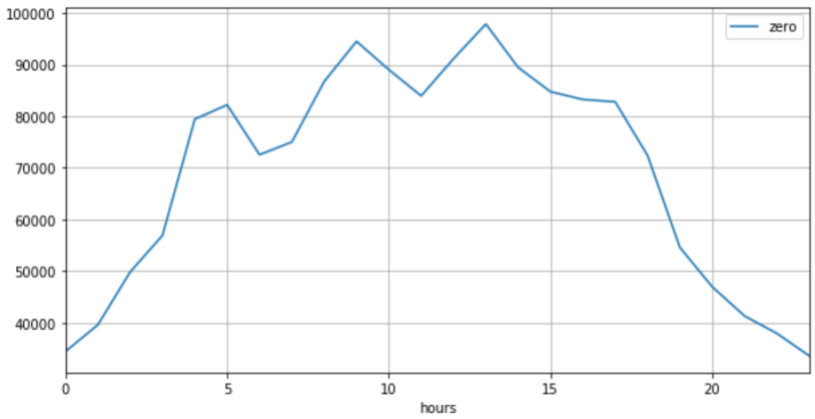
The plots above give us a basic understanding of what values are present in the dataset and how they are distributed. For example, C21 feature has a distributed value ranging from 0 to 250. The values are more concentrated between 0 and 100. Similar is the case with C14 and

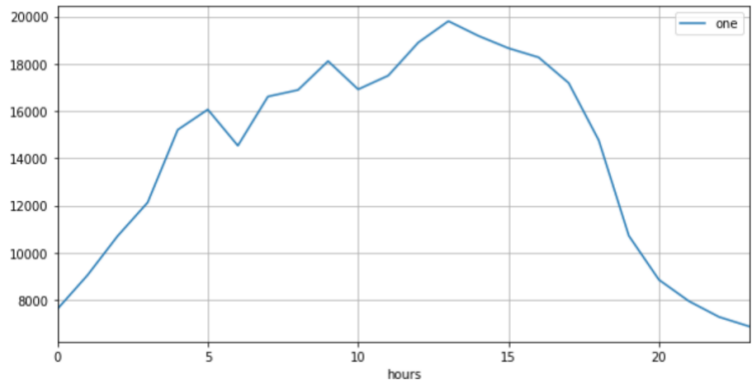
C17 the values are distributed from 0 to 25000 and 0 to 30000 respectively. And for both these features these values are concentrated in the latter half of the graphs. It is a little difficult to interpret the results without knowing what these features mean or indicate. This is one place to start, we can calculate the CTR for each of these to see if they have any effect on the click prediction.

	zero	one	total	CTR
0	34444	7619	42063	0.181133
1	39595	9050	48645	0.186042
2	49834	10707	60541	0.176855
3	56928	12128	69056	0.175626
4	79439	15208	94647	0.160681
5	82203	16067	98270	0.163499
6	72547	14535	87082	0.166912
7	75001	16616	91617	0.181364
8	86712	16894	103606	0.163060
9	94491	18116	112607	0.160878
10	89011	16920	105931	0.159727
11	83940	17502	101442	0.172532
12	91129	18904	110033	0.171803
13	97846	19809	117655	0.168365
14	89440	19184	108624	0.176609
15	84741	18661	103402	0.180470
16	83246	18275	101521	0.180012
17	82783	17190	99973	0.171946
18	72407	14762	87169	0.169349
19	54604	10719	65323	0.164092
20	46968	8857	55825	0.158657
21	41317	7948	49265	0.161332
22	37937	7282	45219	0.161039
23	33602	6882	40484	0.169993

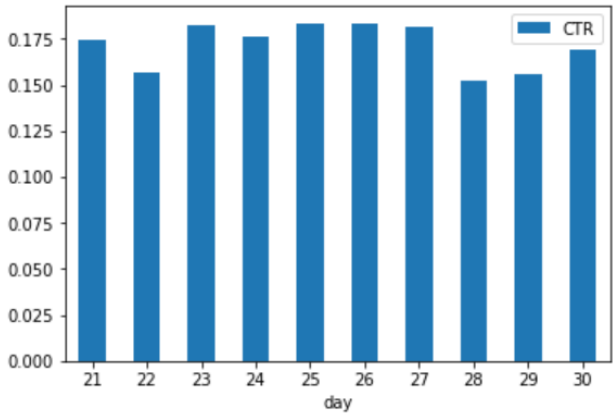


We grouped the hours and clicks for each day and calculated their CTR. The graph below shows the trend according to hours of the day. For both clicked (1) and non-clicked (0) the values were higher during mid-day as compared to Early morning and late at night. This makes more sense as there are more people online in general during the day. But If we see the CTR values the highest value is at 1AM in the morning. Indicating that out of the total people online at that hour, most of them will click on an ad.

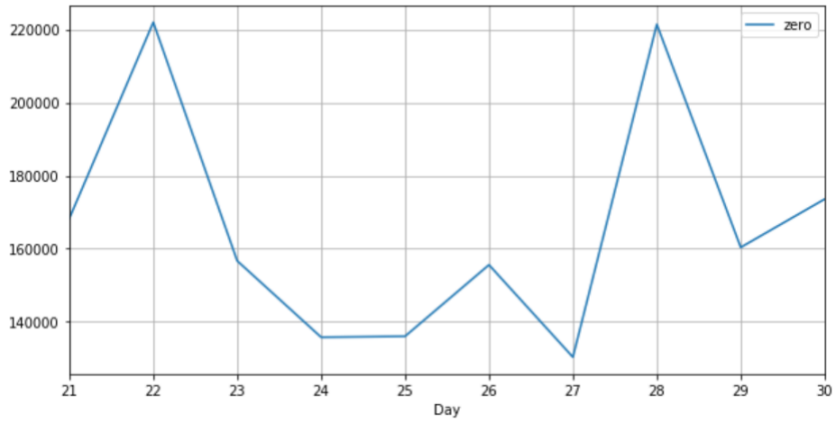


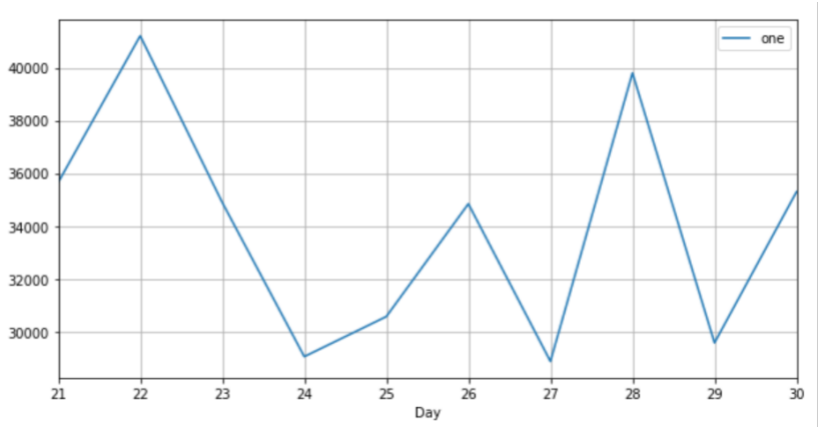


Day Parameter:

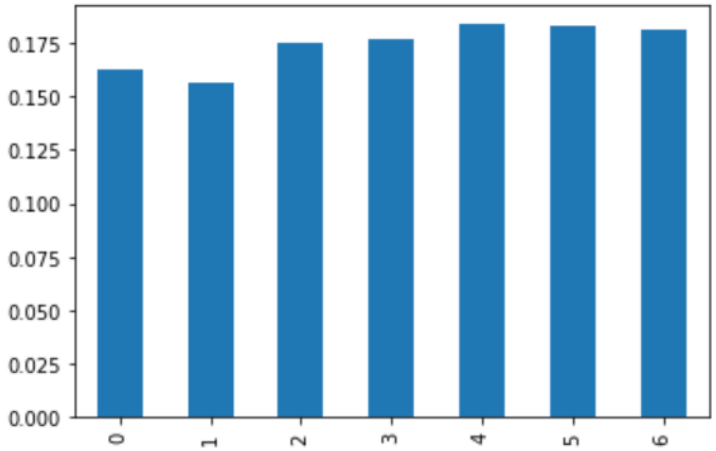


To understand the Day parameter, we sum the number of clicks per day and then calculate their CTR. And the CTR for Day 25 was found to be the highest. There could be a number of factors affecting this, it could be that it was a weekend or that there was a festival coming up, or the company had some sales or promotions going on. There could be a number of factors that can lead to this result. Let's look at the count of the clicks to better understand.

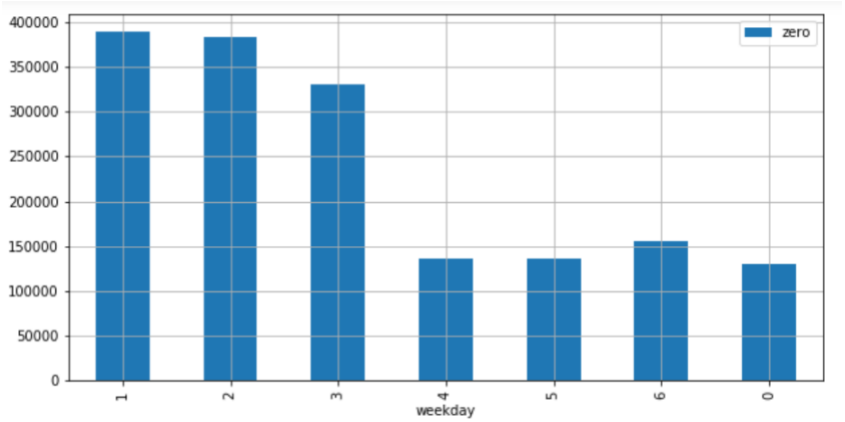


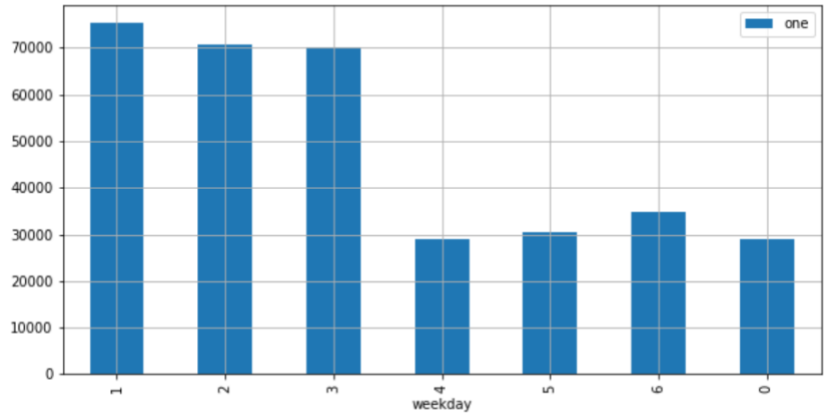


It can be deduced from the two graphs above that the count of clicks (1) and non-clicks (0) were both very low on Day 25 which could be another reason why the CTR was the highest. In comparison to the other days on day 25 out of the total people online majority of them ended up clicking on the ad hence giving us a higher CTR value.



So far, we covered the trends for hour of the day and day of the month next we see the trends for the week. The 4th day of the week had the highest CTR values closely followed by the 5th and 6th day.





The counts are in contract and show a similar trend like Day of the month. When the counts are less for both clicked (1) ads and non-clicked (0) ads the CTR is usually higher.

Discussion

Since Ad clicks contribute to major revenue generation of companies like Google, it was an interesting topic to research upon and build models. The performance would have been a lot better if there were more columns captured in the dataset like the search query, user information and location. From our research for our dataset, logistic regression model would generate better log-loss had the data been linearly separable. Since it was not linearly separable, gradient boosting model was applied and it gave a low log loss value.

We could have built a calibrated model on our base model and carried out logistic regression to try to fit to better with our dataset. In future, we plan to take this assignment forward in big data and use a better dataset with more columns capturing important aspects missed in this dataset. More columns would help in better logistic regression model formation.

References:

(Aug 19, 2018). Retrieved from- Applied AI https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4143/live-session-on-ad-click-prediction/7/module-6-machine-learning-real-world-case-studies

(Sept 19, 2018). Retrieved from- Kaggle Competition - https://www.kaggle.com/c/kddcup2012-track2/overview/description

Retrieved from- GeeksforGeeks https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/

Retrieved from - Having an Imbalanced Dataset? Here Is How You Can Fix It - https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb

Retrieved from - No need to remove multicollinearity as not affects predictions https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/

Retrieved from -Logistic Regression Analysis https://www.statisticssolutions.com/assumptions-of-logistic-regression/

Retrieved from - Encoding cyclical continuous features - 24-hour time https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/