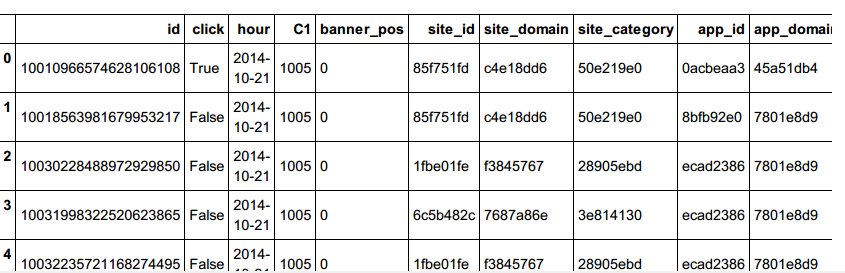
# EXPLORATORY DATA ANALYTICS

**Reading the data**

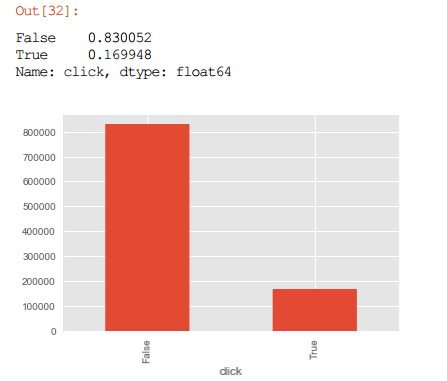
This is just a little snippet from the head of the data. We have 24 features spread across 1 million records.



ILOC statement on the sampled training data

**####Preliminary CTR analysis####**

Checking the click-through rate behavior in the initial stage just to calculate the CTR before we dive in.



This effectively is a click-through rate of ~17%.

The **TRUE**values represent the entries where the ads were clicked and that rounds nearly to 17%. Hence, we can say that about 17% of the ads were recorded as clicks from the sample of 1 million records.

**Feature Engineering**

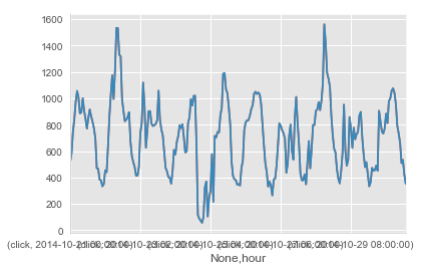
[Andrew N](http://www.andrewng.org/)g says — “ Coming up with features is difficult, time-consuming, requires expert knowledge. “Applied machine learning” is basically feature engineering.”

Keeping in mind the need for highly scalable and powerful systems/machines and the restrictions I was operating under for this particular project, my extent of performing feature engineering went just as far gaining some knowledge from the available features; uncovering insights/patterns and garnering some sort of a lead from that to later come up with an appropriate model to forecast the click probability.

To begin with, we assess the ‘Hour’ metric.

**HOUR**

To study the click and impression trends w.r.t different hours of the day, we plot the clicks and impressions w.r.t different hours of the day. HOUR metric is recorded as timestamps. The maximum number of impressions generated around 1 P.M ~ so that’s a valuable insight.



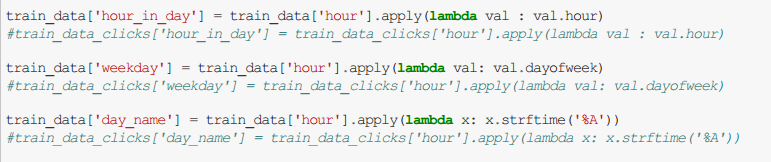
Using Matplotlib to plot click trends w.r.t Hour

**Introducing new attributes**

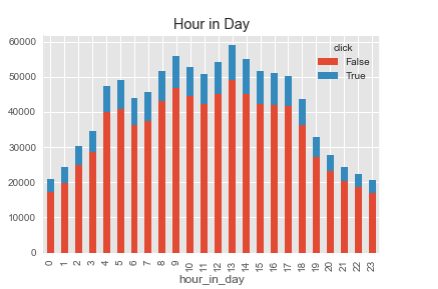
Little do we know about the user behavior across different days of week or say during different hours of any given day. Hence we go ahead and introduce a couple of new attributes by making use of the ability to parse the available ‘hour’ timestamps.

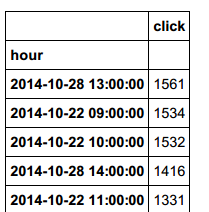
**Creating hour\_in\_day, weekday/day\_name**

So we create 3 new attributes in hopes of being able to visualize the usefulness of the HOUR metric on a deeper level and to get more useful insights.

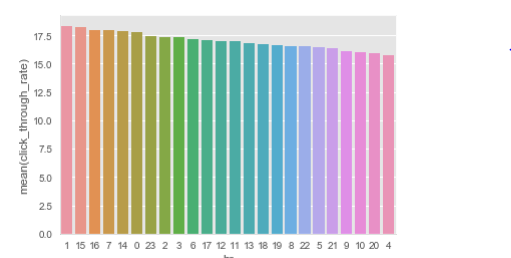


**Hour\_in\_day**





*1 P.M seems to be the time around which most clicks are evident. This window is usually a representation of lunch hours*



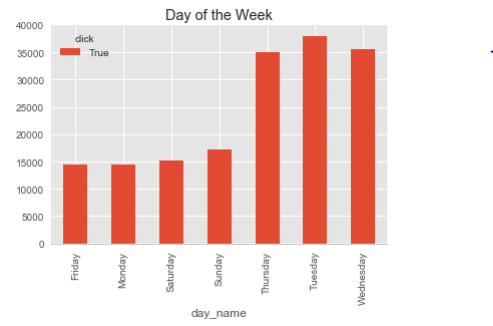
Using Seaborn to visualize CTR spread across different hours of the day

Next up, we visualize day\_name as an attribute.

**DAY\_NAME**

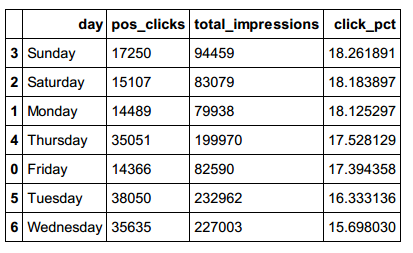
https://cdn-images-1.medium.com/max/1000/1*3uppWT0rBEHk-BYJPNE3hg.png

This gives us an idea about the click-spread across different days of the week.



Click spread across different days of the week

It can be seen that most clicks occur on Tuesday, followed by wednesday and Thursday. In that order. In addition to this, we can also perform a day-wise analysis of click-through rates.

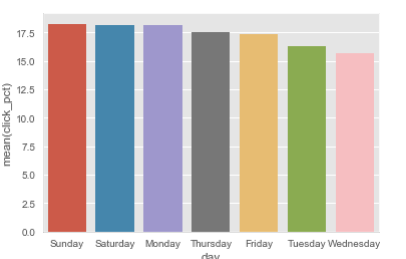


Click-through rates across days of week

**pos\_clicks** — count of total clicks on that particular day

**total\_impressions** — total number of impressions across that day

**click\_pct** = 100\*(pos\_clicks/total\_impressions) ~ click-through rate on different days of the week



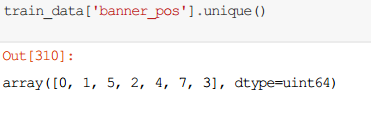
Using Seaborn to generate a bar plot to depict the click distribution across different days of the week.

It’s visible that Sunday is when we get the highest click-through rate. Logically this does make sense because Sunday offers more leisure time for the visitors to browse the applications thus increasing the chances of getting clicks which are clearly reflected by the visualization.

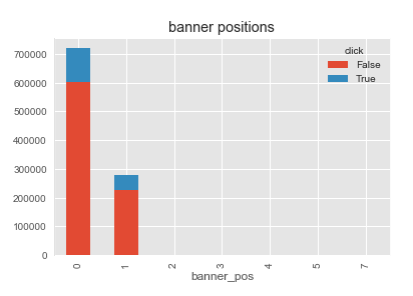
**BANNER\_POSITION(attributed as banner\_pos)**

Banner positions representing attractive and appealing designs that might highly affect a user’s behavior and in turn trigger their decision to click. Or not. Hence making it an effective metric to predict clicks

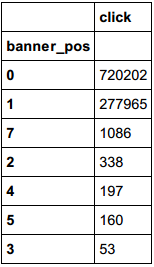
Evidently, there are 7 different banner positions. The 7 banner positions might represent ad placing on a given web-page or like the positioning of advertisements



Banner positions



It can be seen that positions 0 and 1 are the most prominent banner positions and have garnered the most number of impressions.



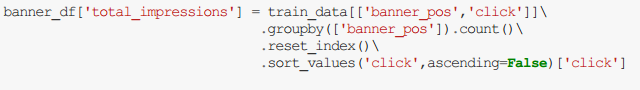
Clicks with respect to different banner positions

**CTR analysis on the banner position**

All we do is define a new data frame with 3 columns — Position, clicks, total number of impressions.

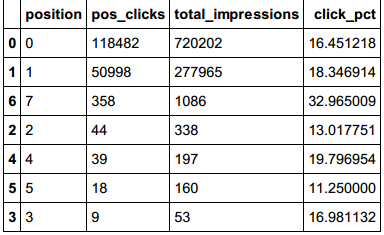


Defining the clicks column in the newly introduced dataframe



Defining the total number of impressions

Finally , a simple ratio of the pos\_clicks with the total\_impressions will give us the click through rate and we append that in a new column as can be seen below.

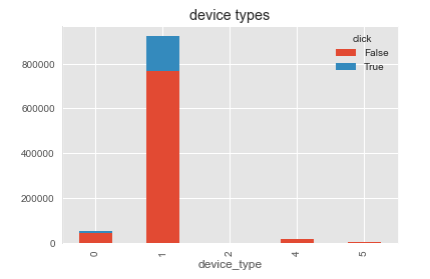


The newly defined dataframe

This tells us that the 7th banner has the highest click-through rate among all the available banner positions.

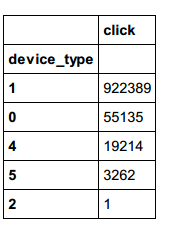
**DEVICE TYPE METRICS**

We’re not entirely sure as to what these device types are so the best we can do is draw assumptions from the click behavior that we see through these visualizations



Distribution of clicks across different device types

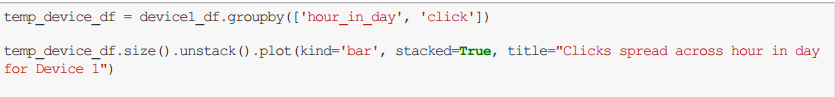
Insight: It can be seen that device type 1 get the most impressions among the 5 devices that are listed.



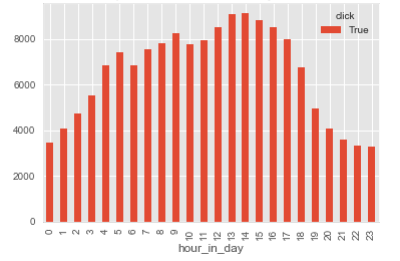
So now we know that device type 1 gets the most number of clicks as well as the most number of impressions.

This could possibly be a cell phone as those number of clicks is humongous so that sounds like a reasonable assumption.

Moreover, we can dive deeper and study the user behavior across the Device 1. That will give us a better understanding of the click spread across say different hours of the day for device 1.

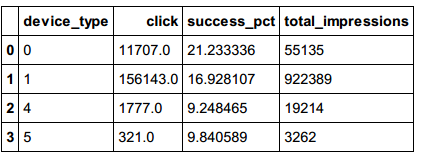


We can reasonably assume that the device type 1 might be a cell phone since the businesses might not prefer showing ads later in the evening which is something that can be seen from the plot below. The click spread more or less is on an incline between the 9 to 5



**Click through analysis with respect to device types**

A merging of two separate data frames was done in order to get to this particular data-frame. The details can be found in the code.

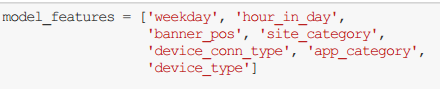


This shows that Device type 0 is the one that actually has the highest click through rate(**21%)**

Because of lack of sufficient information on app related metrics and C14-C21(anonymized categorical variables), the scope for performing an extensive EDA wasn’t as wide. Hence, we proceed with the 2nd stage where we develop the prediction model.

**Stage 2: Developing the prediction model**

We start of with defining a set of model features that we believe will best help us in coming with an optimized model. Features that will make the most sense, so to say.



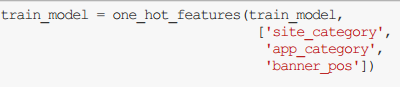
Model features

Setting up the training model clubbing the model features with the target metric that is the click column. We take a 10% fraction of the original training sample so that is 10% of 1 million i.e 1 lakh records. This is done in order to fasten the computation.

https://cdn-images-1.medium.com/max/1000/1*U_nmUvhH9qlDWx-lTMXv6A.png

**Using one hot notation to make hashed attributes readable**

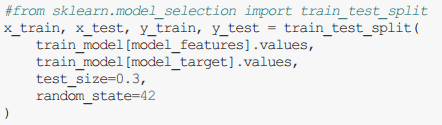
Attributes like site\_category and app\_category are hashed and need to be presented in a readable format. One way to deal with this was to use a one hot notation to achieve the same.



Using one hot notation

Once we have the training model ready, we take an instance of it where we filter out the target column and keep just the features

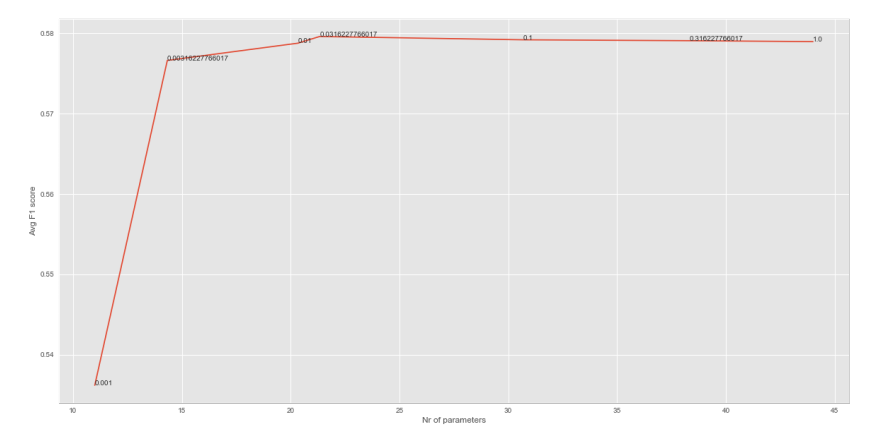
**Defining the train-test split using sci-kit learn**



**Feature selection**

Feature selection is performed to reduce the dimensional space occupied and to deal with overfitting. Grid Search cross validation is used to perform the same and L1 regularization is used to obtain a trade off between the number of features and the F-1 accuracy score.

F-1 score is used as a performance metric because it represents a harmonic mean between the precision and recall values.

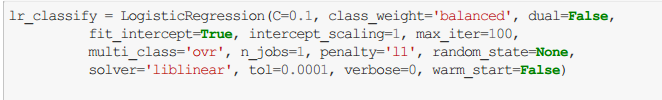


Logistic Regression with L1 regularization and balanced class weights

Once we are done defining the Logistic regression model using L1 regularization, we can see that parameters obtained using c=0.1 manage to reduce dimension and optimize the dimensional space which optimizes the execution time. This in turns also improves the generalization capacity.

**Formulating the Logistic Regression classification equation**

Tuning hyper-parameters involves tweaking maximum iterations, number of thread jobs, class weights and various other metrics.



Logistic Regression equation

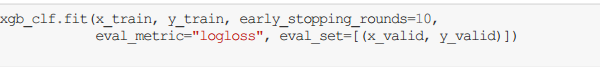
**Stage 3: Evaluation metrics (A Gradient Boosting Approach)**

Using the xgboost module and classification reports from sci-kit learn to assess the accuracy of the models

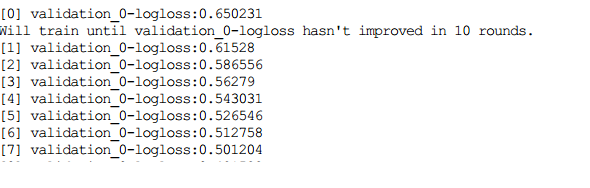


The essential goal of the model is to minimize the log loss values. Log loss estimates reflect the performance of the model with respect to the prediction label. These labels lie between 0 and 1 and are the target variable.

We define the classifier and fit it with the x\_train and y\_train splits.



What follows is a series of iterations where log-loss values are captured with every iteration. It looks something like this:



100 rounds ~ 100 iterations

Even though there are 100 iterations, the loss loss stops improving past the 10th round. There’s a drastic improvements in the first 10/12 rounds. But the drop reduces in the subsequent rounds.

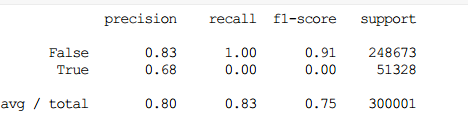
Now that we have a log loss estimate, we can go ahead and predict the labels/data points

https://cdn-images-1.medium.com/max/1000/1*FJtfemjQsO6V1e_SN3G99g.png

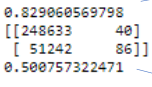
Predicting the labels/probabilities

**Evaluating the metrics**

As it can be seen, the model correctly predicts the non-clicks cases 83% of the time on aggregate and it correctly predicts the clicks 68 times on 100. This varies as samples keep getting randomly drawn as fractions.



Classification report



Various metrics

The model runs with 83% accuracy. The confusion matrix records the numbers of true positives/negatives and false positives/negatives

**0.50075** is the area under the receiver operating characteristic curve and it implies the expected position of the positives before drawing a uniformly drawn random negative