Outbrain Click Prediction

Final Project: Advances in Data Sciences

**Under the guidance of Professor Srikanth Krishnamurthy**

**Team 8**

**TABLE OF CONTENT**

1. [About Outbrain](#_About_OutBrain)…………………………. ………………………………………………………………………………. 3
2. Team Members and contribution ………………………………………………………………………………. 4
3. Problem Statement ……………………………………………………………………………………………………. 6
4. [Overall Flow](#_OVERALL_FLOW) ………………………………………………………………………………………………………………. 7
5. Luigi Workflow …………………………………………………………………………………………………………... 8
6. [Data Pre-Processing](#_DATA_PRE-PROCESSING) …………………………………………………………………………………………………… 14
   1. [Data Sources](#_DATA_SOURCES) ……………………………………………………………………………………………………….. 14

6.2 [Data Scraping](#_DATA_SCRAPING)  …………………………………………………………………………………………………….. 15

1. [Data Exploration](#_DATA_EXPLORATION) ………………………………………………………………………………………………………… 17 7.1 [Analyis on Python](#_Analyis_on_Python)…………………………………………………………………………………………………. 17
   1. [Analysis in Tableau](#_Analysis_in_Tableau)……………………………………………………………………………………………….. 41
2. Data Processing and Modelling…………………………………………………………………………………. 45  
   8.1 File Selection………………………………………………………………………………………………………… 46
   1. Data Validation and Cleaning ………………………………………………………………………………. 47
   2. Metadata Modelling ……………………………………………………………………………………………. 49

9. Indiviual Advertisement Probability…………………………………………………………………………… 60

10. List of AD’s for a particular display in decreasing order of click probability……………….. 77

11. References………………………………………………………………………………………………………………... 93

# **About OutBrain**

Outbrain pairs relevant content with curious readers in about 250 billion personalized recommendations every month across many thousands of sites

It uses [behavioral targeting](https://en.wikipedia.org/wiki/Behavioral_targeting) to recommend articles, slideshows, blog posts, photos or videos to a reader, rather than relying on a more basic "related items" widget. The sites with the recommended articles pay Outbrain for this service, and Outbrain pays the site on which the links appear.

1. **TEAM MEMBERS AND CONTRIBUTION**

1. Antriksh Antriksh

NUID : 001698913

Linkedin Profile URL : https://www.linkedin.com/in/fnu-antriksh/

1. Sweta Bajaj

NUID : 001643792

Linked Profile URL : <https://www.linkedin.com/in/swetabajaj/>

1. Yogita Jain

NUID : 001643815

Linked Profile URL : <https://www.linkedin.com/in/yogita-jain/>

CONTRIBUTION CHART

1. **PROBLEM STATEMENT**

Millions of user use social network for surfing, visiting countless websites and clicking on countless ads/recommendations on these website.

Knowing what the users are interested in and what the users are using in real world would be of great significance for future recommendations used by marketing team to attract potential users as well as ad placements and real time bidding

1. Predicting the likelihood of users clicking on a particular content
2. Ranking the recommendations in each group by decreasing predicted likelihood of being clicked

# **OVERALL FLOW**

Flowchart of the Steps Performed

DOWNLOAD DATA FROM KAGGLE/DATA SCRAPING

DATA EXPLORATION ON PYTHON

DATA EXPLORATION

DATA EXPLORATION ON TABLEAU

DATA SET SELECTION

DATA CLEANING/VALIDATION

METADATA MODELLING

JOINED FILES CREATED(TWO)

Run All models on Jupyter

Run All models on Jupyter

AD Id’s in decreasing order

Indiviual Advertisement Probability

Select Best Model

Select Best Model

Best Model Run on Azure

Best Model Run on Azure

Web API creation

Web API Creation

Create Webapp using R SHiny

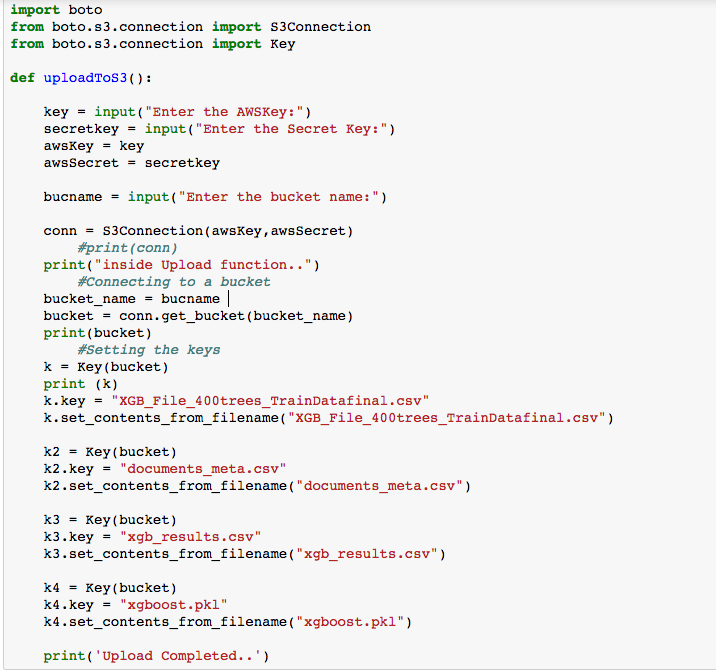
Deploy Web App

# **Luigi Work Flow**

## Code to upload to S3

Below is the code to upload the trained model, test data, train data, prediction file. At first we need to make a connection to the AWS using aws key, secret key and bucket name.

Once connection is established it will move the data to s3 bucket.



**Luigi** is a Python module that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management. We have used Luigi for managing our workflow to download data from the Kaggle website, creating the metadata model, creating trained model, test model and finally uploading the model with test and train data to aws s3 bucket.

The four py files used for Luigi script are:

Download\_OutBrain\_Data.py

MetaData\_For\_Training.py

Upload\_To\_S3.py

Outbrain\_luigi\_Script.py

PYTHON CODE



Once the tasks are completed successfully, the above mentioned files will be uploaded to S3 bucket. Below is the success code for running the Luigi workflow.



# **DATA PRE-PROCESSING**

Data was downloaded from the kaggle website through python script and subsequently used for data exploration to see how the data looks like, before it could be used for model creation.

# **6.1 DATA SOURCES**

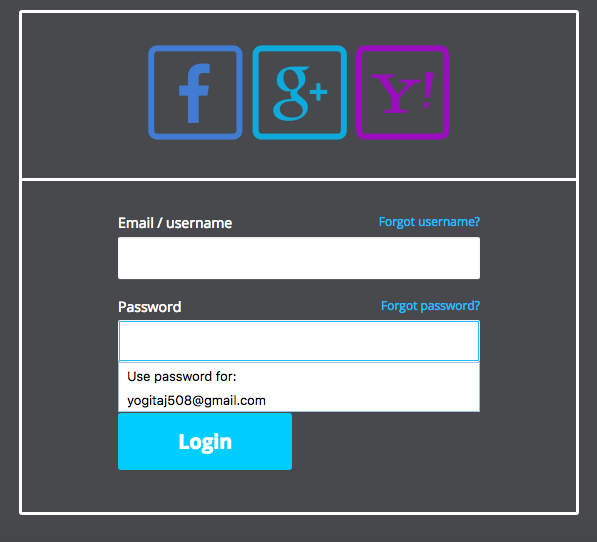
We used requests python library to scrape data from Kaggle.com. Before downloading the data, it requires to log in. Passed the credentials to the script completed the downloading of the data. There are 11 zip files that were downloaded

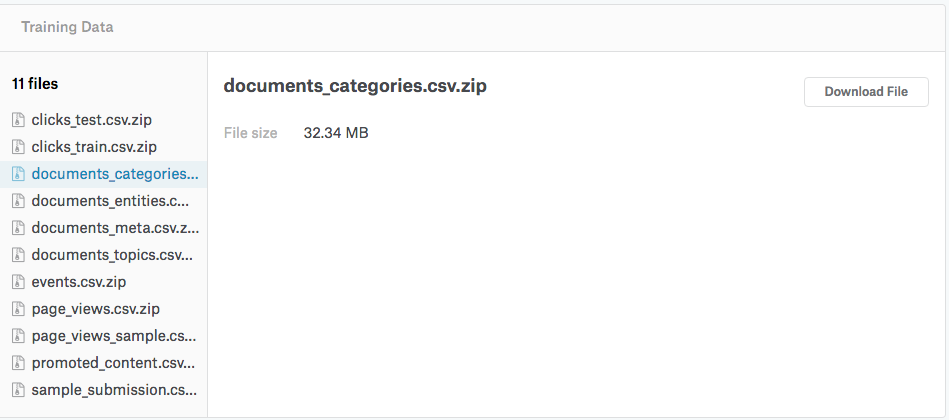
Files:

* Clicks\_test.csv.zip
* Clicks\_train.csv.zip
* Documents\_categories.csv.zip
* Documents\_entities.csv.zip
* Documents\_meta.csv.zip
* Documents\_topics.csv.zip
* Events.csv.zip
* Page\_view.csv.zip
* Page\_views\_samples.csv.zip
* Promoted\_content.csv.zip
* Sample\_submission.csv.zip

# **6.2 DATA SCRAPING**

**Log in Page:** Below is the screenshot of the login page that we will start scraping from and land on data download page as shown below.





PYTHON CODE

The below function in python code will download the files from the kaggle website after successful log in.

****

# **DATA EXPLORATION**

## **7.1 Analyis on Python**

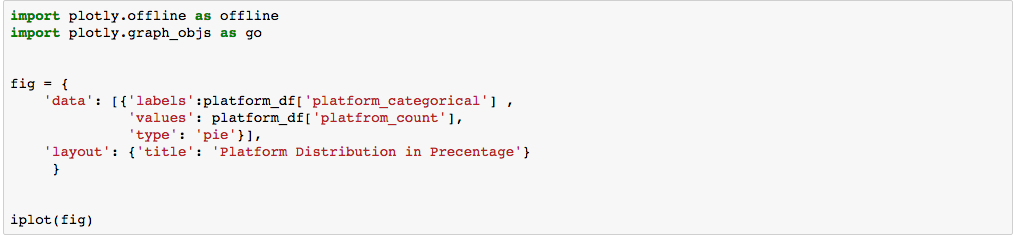
**Following are the analysis that were done on events data to look at specific insights**

1. **Analysis of percentage of clicks across different platform’s in events table:**

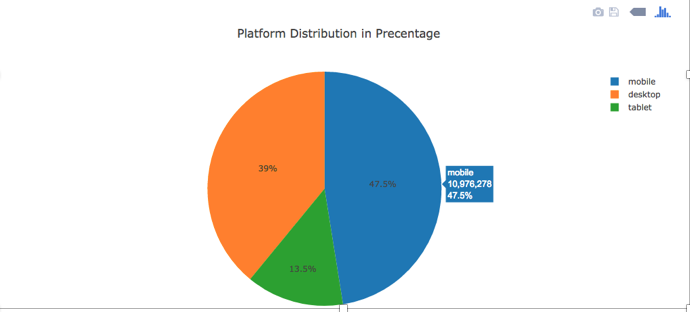
PYTHON CODE

Since platform was numeric, converted it to categorical column as below.

Screen%20Shot%202017-04-27%20at%2010.21.47%20AM.png



GRAPH



OBSERVATIONS

* Maximum percentage of clicks were made through mobile phones, followed by desktop and then tablets. Mainly because:
* App Availability: the app is available that is available on desktop is now present on model too.
* Convenience: Games or social networking apps frequently serve as a way to pass the time while on the subway commuting home or in a cab or surfing net. This directly reflects the increase in uses of mobile devices

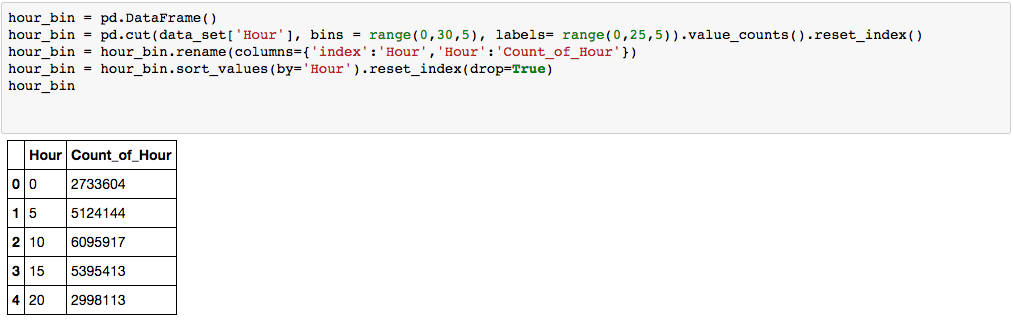
1. **Analysis of click frequency during different hours of a day in events table.**

PYTHON CODE

The events table has a column timestamp, which is a Unix timestamp. We converted that to hours and days using the standard formula and created a new hour column as shown below.

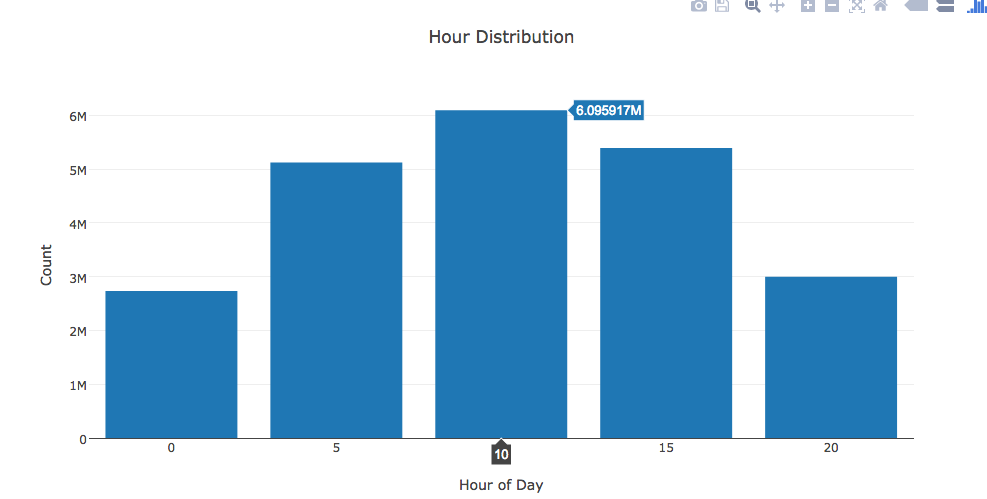
**Screen%20Shot%202017-04-25%20at%2011.41.01%20PM.png**

Next, we divide the hours into 5 bins with a difference of 5 between them and found count in each hour and added to the bin as shown in the below code.

****

****

GRAPH

****

OBSERVATIONS

* We observed that the click frequency is very high between 10-15 hrs. Around 10-11 a.m. time people are usually on their commute to work, school etc. and pass their time surfing net. Around 1-3 p.m. is usually lunch time where people get time to surf internet and that’s the frequency is high.
* The frequency in the bucket 15-20 is again when people are traveling back home and around dinner time when they have time to surf internet.

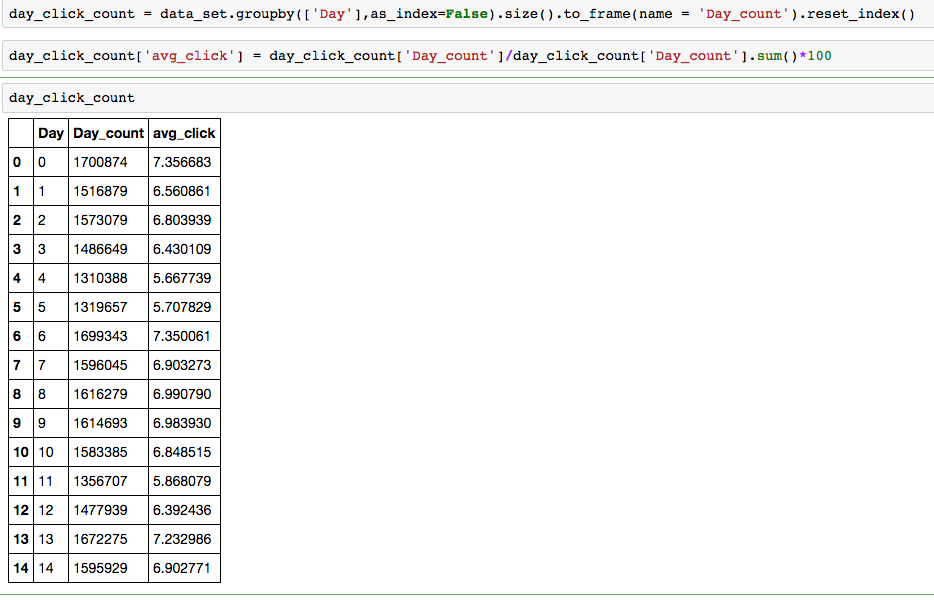
**c) Analysis of average click frequency on different days in events table**

PYTHON CODE

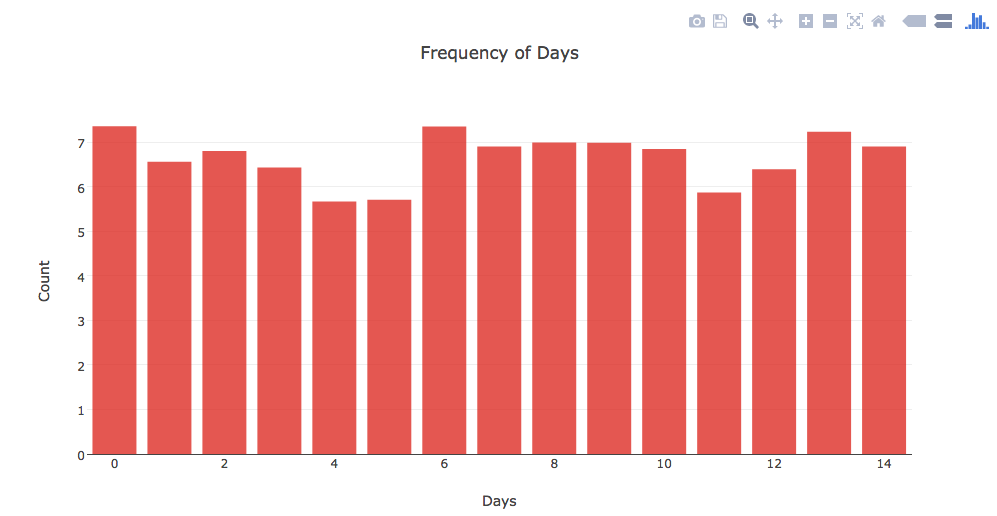
The dataset contains data for only 14 days. Hence we will see the distribution of click frequency on these 14 days. We extracted day from the timestamp column and created a new day column.

Screen%20Shot%202017-04-26%20at%2012.00.25%20AM.png

Next, we calculated the frequency of each day and using that calculated the percentage of occurrence of each day.



GRAPH



OBSERVATIONS

* The 0th day which is June 14th has maximum numbers of average clicks compared to other days as it was US Flag Day.
* The 6th which is June 20th has second highest number of average clicks is the Summer Solstice marks the beginning of the summer season in the Northern Hemisphere.

**Following are the analysis that were done on Page views data to look at specific insights**

Page views is basically a log of users visiting the documents.

1. **Analyze the frequency of page views across different platforms**

PYTHON CODE

There are three platforms: desktop, mobile and tablet. The dataset contains a platform column which is

numerical so we converted into categorical column by using a map function as below.

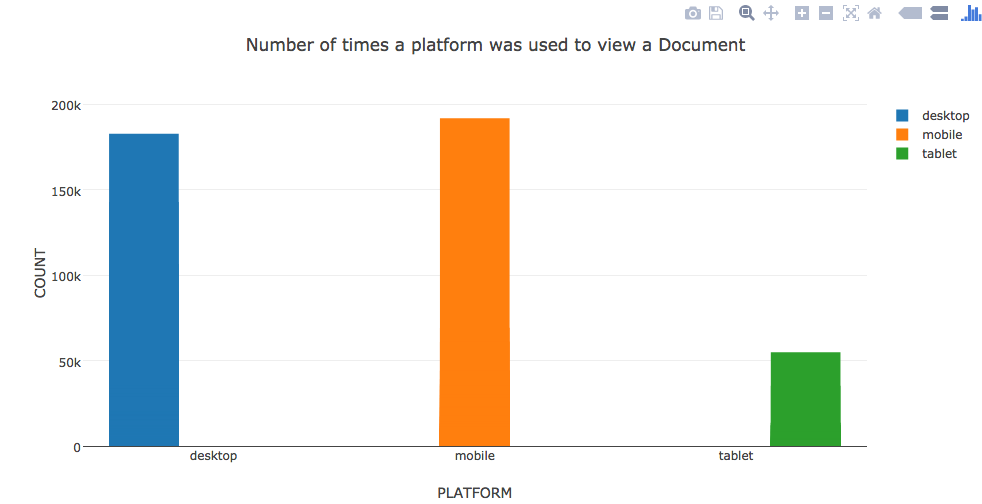
Screen%20Shot%202017-04-26%20at%2012.26.30%20AM.png

Next, we calculated the frequency of each platform group by display\_id and then created three separate data frames for each platform type to draw a plot.





GRAPH



OBSERVATIONS

* The number of views to page or a document online are highest through mobile, followed by desktop and then tablet.
* Again the main reason is people mostly go online when they are travelling and mobile being the most convenient device.

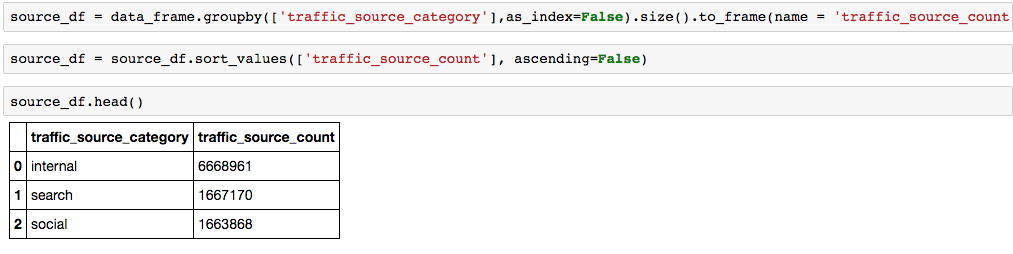
1. **Analyze how traffic sources is related to accessing a page online**.

PYTHON CODE

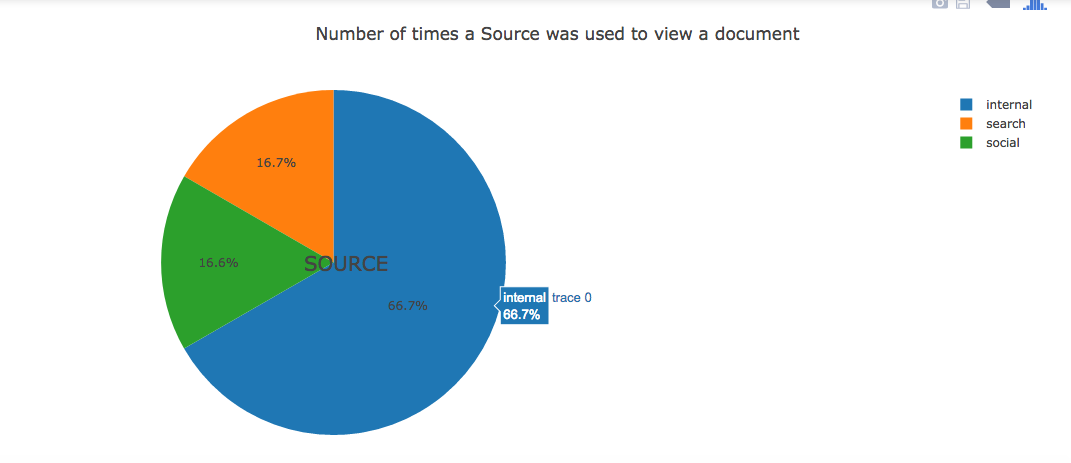
There are three types of traffic sources in the document. Since they were numerical we converted them to categorical



Next, We calculated the frequency of occurrence of each traffic source.



GRAPH



OBSERVATIONS

* Internal traffic source was about 66.7 % which is more than half of the traffic.

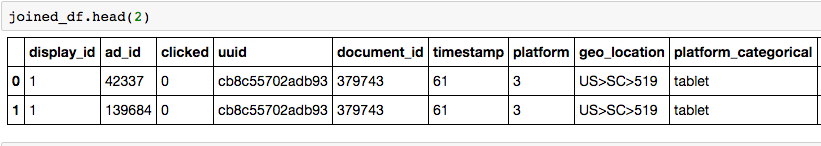
**Following are the analysis done on events and clickstrain data to look at specific insights**

1. **Analysis of number of clicks by hour of day in the events table and clicks\_train table.**

PYTHON CODE

For this analysis we did an inner join on the two datasets: events and clicks\_train on display\_id.

Screen%20Shot%202017-04-26%20at%209.56.49%20AM.png



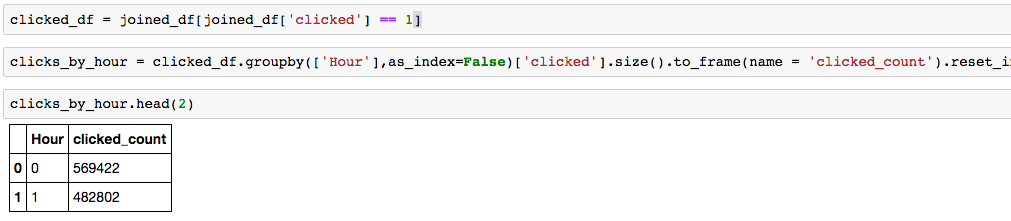
Since the timestamp is in milliseconds, we calculated day and hour using the below formula.



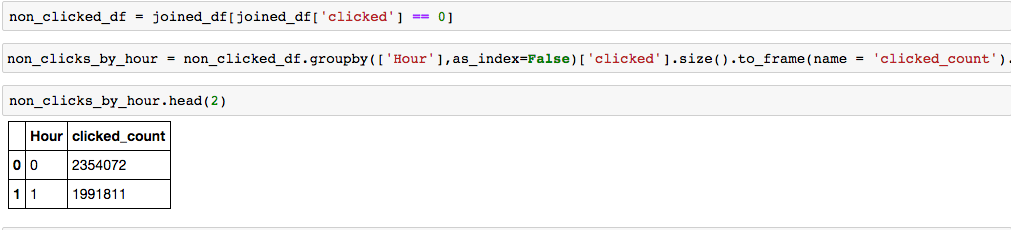
Next, created two separate data frame for click and no-clicks based on the condition:

(joined\_df[‘clicked’] == 0 or joined\_df[‘clicked’] ==1]

**clicked\_df**



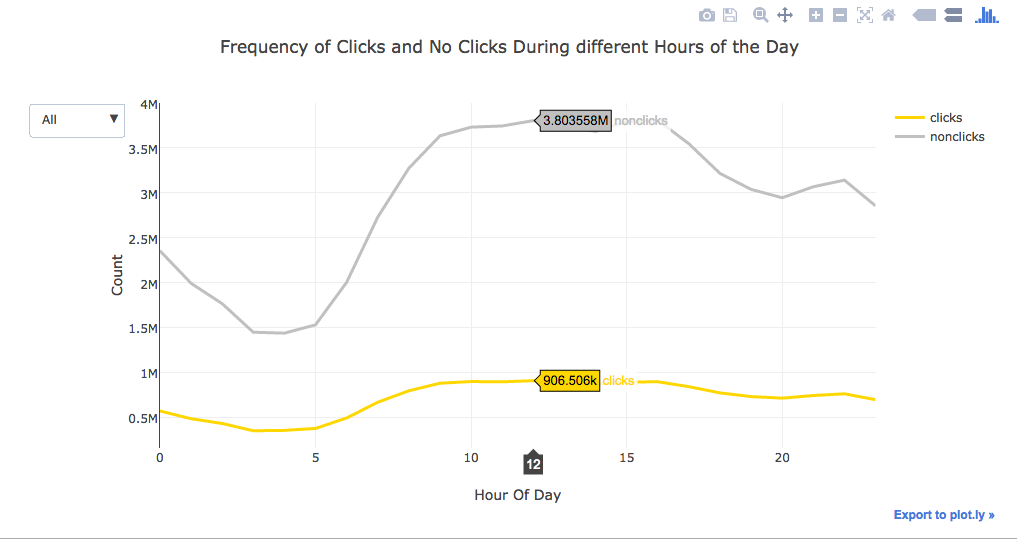
**non-clicked\_df**

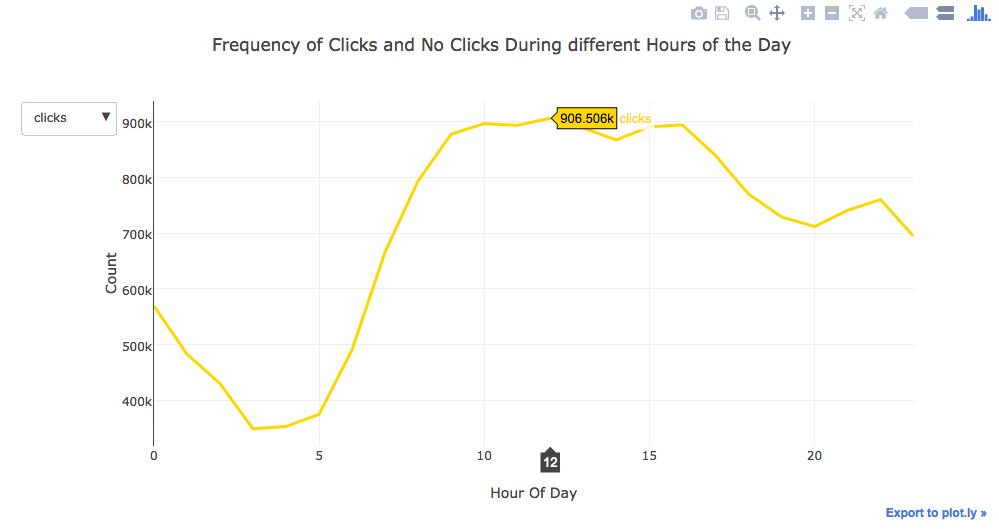


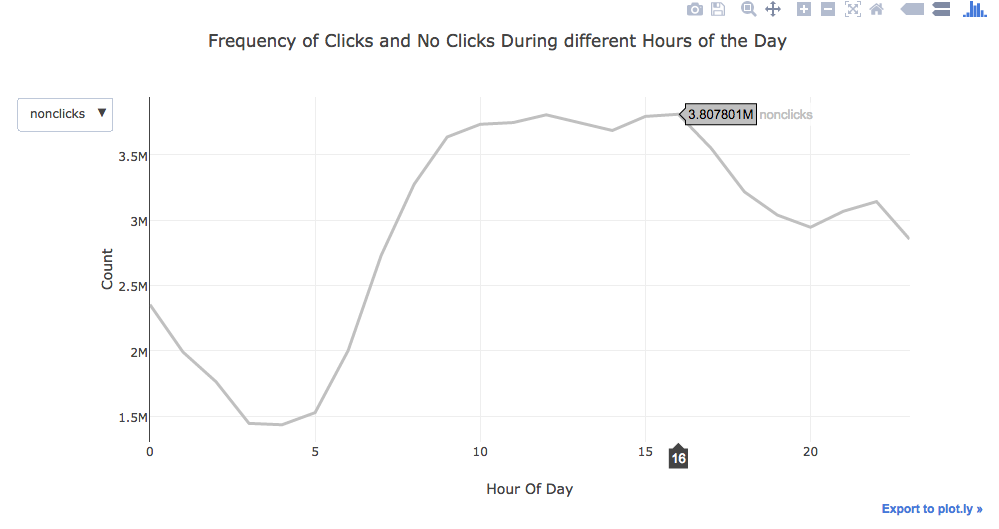




GRAPH







OBSERVATIONS

* The number of clicks increase from 9.am. and reaches highest at around 10 a.m. and remains high till 12p.m. and then gradually decreases till it hits 3.p.m. It again rises from 3p.m.
* The least number of clicks is again

1. **Number of Clicks and no-Clicks on Different Days**

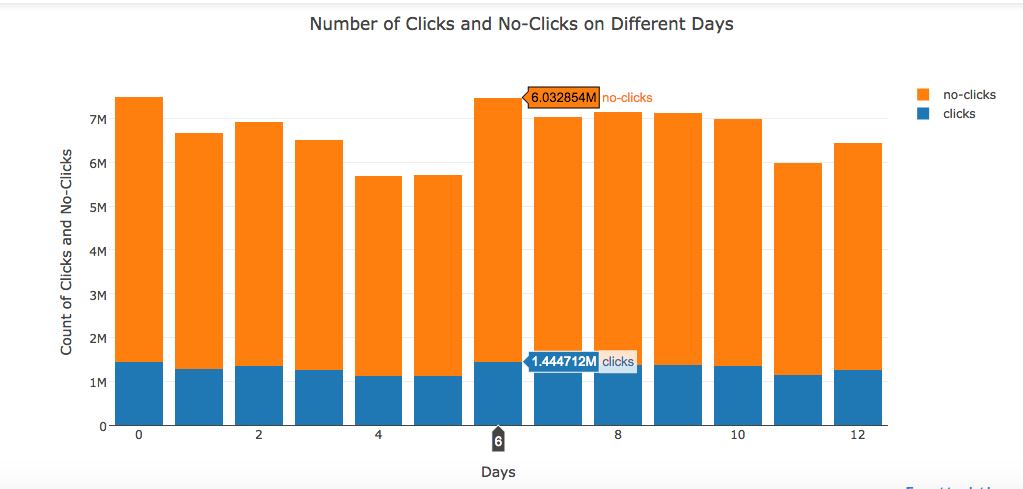
PYTHON CODE

Calculated the number of clicks and no-clicks grouping by day and created two separate data frames as shown below.





GRAPH



OBSERVATIONS

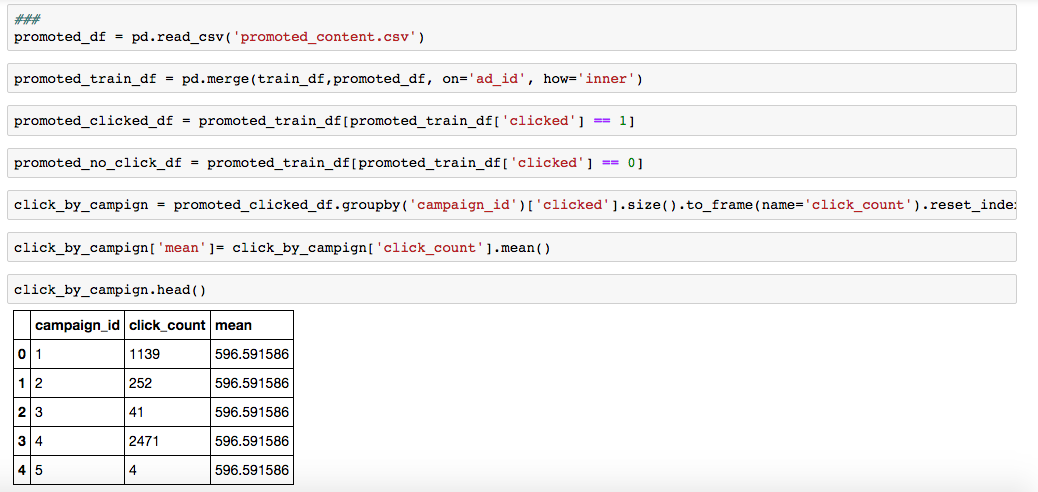
* The number of no clicks on the day 6th ranks the highest.
* The number of clicks remains almost constant during all the 14 days of the data.

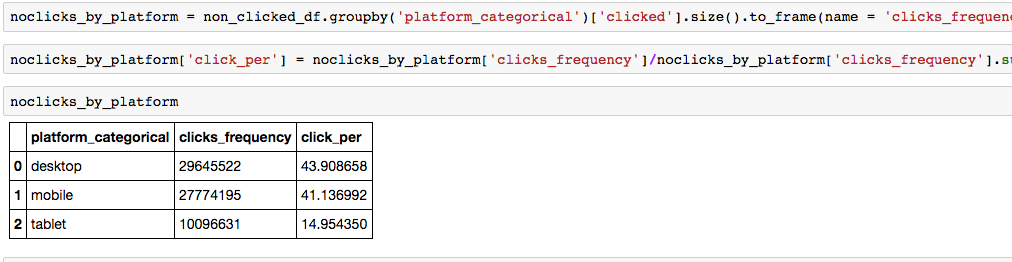
1. **Analyze the Frequency of Clicks for various Ads**

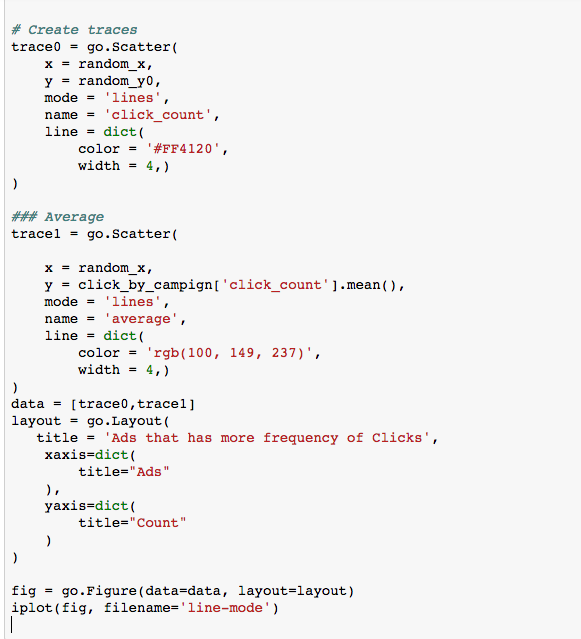
PYTHON CODE

In order to analyze the frequency of clicks across different ads in the dataset we did an inner join promoted\_content and clicks\_train files on ad\_id as below.

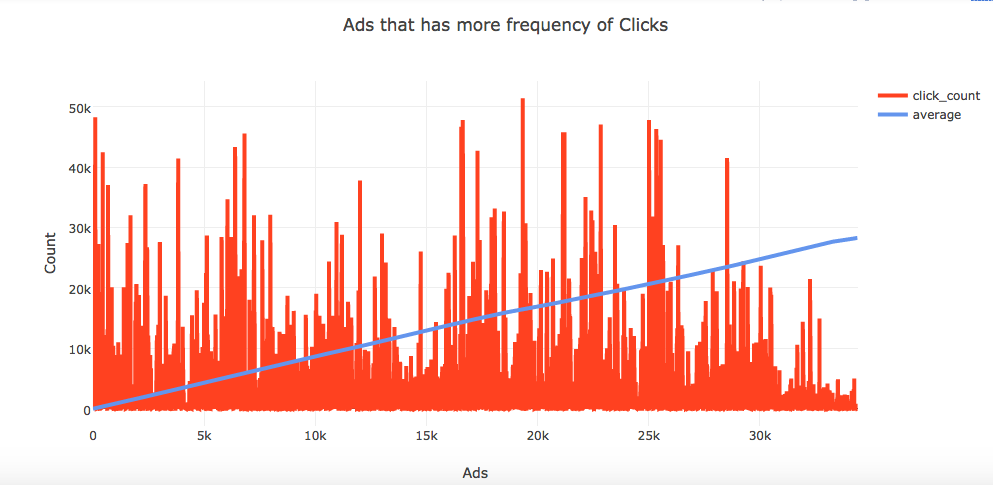
Created two different data frames for promoted content, and one with clicks and one with no clicks.

****

****

****

GRAPH



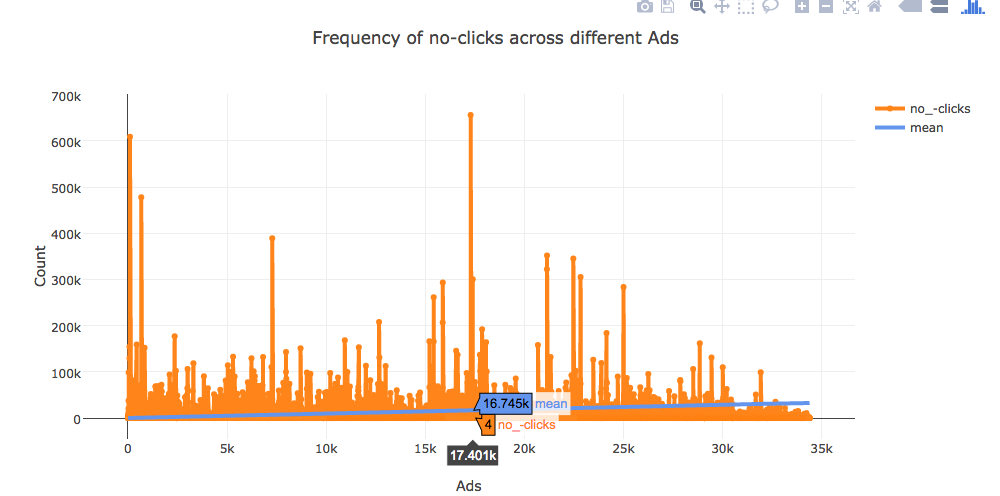
1. **Find number of no-clicks for each campaign id**

PYTHON CODE





BAR CHART PLOT



## **7.2 Analysis in Tableau**

## Geo Analysis 1

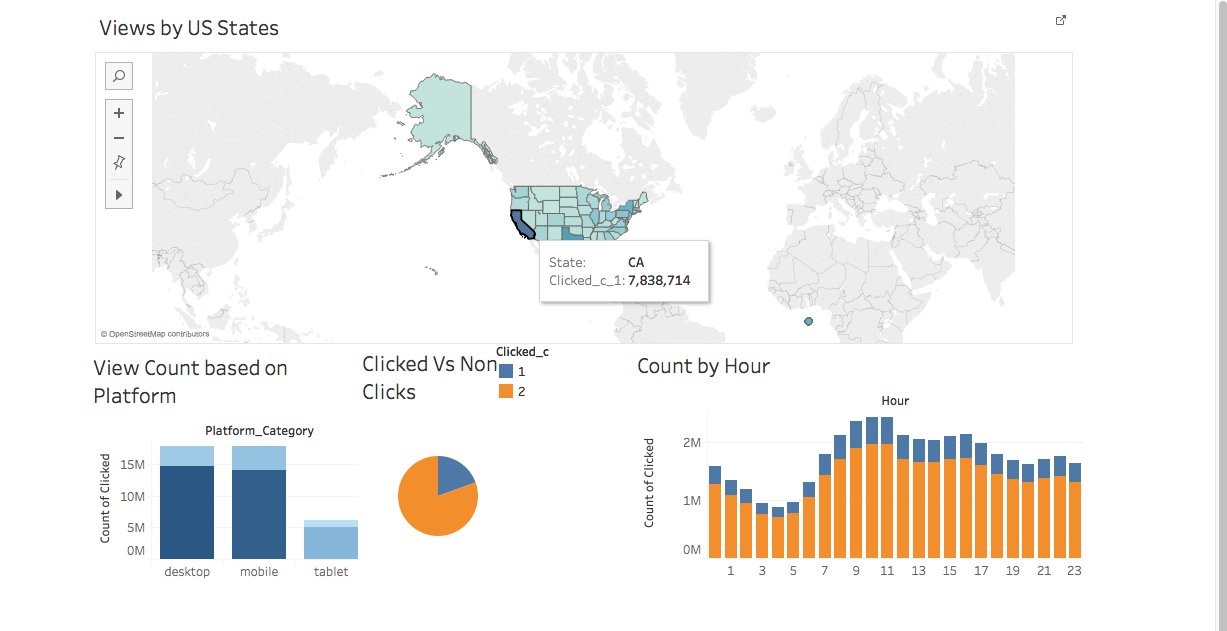
We analyzed the dataset for each country based on different platforms by hour and day to understand the variations in page clicks, no clicks and page view by a user.

OBSERVATIONS

* From the dashboard we can conclude that USA that the maximum number of view rate. The frequency of clicks gradually increases from 6 am and was high around 10 a.m. to 12 p.m.
* The click was highest on Day 0 which is June 14th. It is a US national holiday.

## Geo Analysis 2

After concluding that US had the maximum number of views per page. We further drilled down to analyze the variations in clicks, no clicks and page views in US states.



OBSERVATIONS

* California state has the maximum number of views.
* The maximum number of views were from mobile devices at around 11 a.m.

1. **DATA PROCESSING AND MODELLING**

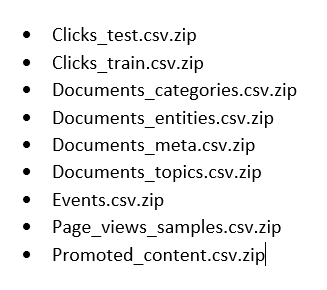
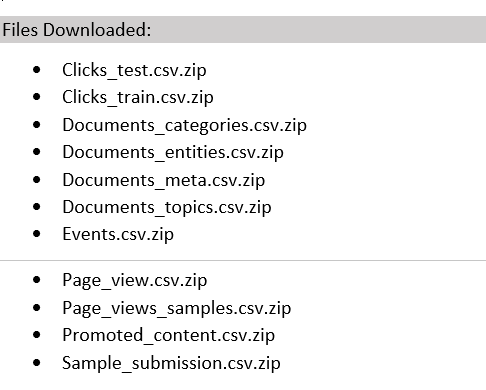
After the files were downloaded from the kaggle websites, following steps were performed.

1. **Data Files Selection**: Based on the models created, relevant data files were considered out of all the downloaded files
2. **Data Cleaning and Validation**: The data in the files was cleaned and validated to ensure clean data is fed for next steps.
3. **Metadata modelling**: As the data is divided into various files, files were joined to create files needed for our model creation.

**8.1 File Selection:**

1. Out of the 11 files downloaded , we took only 9 files for our further analysis.
2. Files Dropped : Reason
3. Page\_view.csv.zip : The file in question had data with the size of around 70 GB and the click\_train
4. Sample\_submission: This is just a sample of how decreasing probablity model output should look like. A similar file will be our output from one of the model flow.
5. The selected files were then taken for further steps.

SELECTED FILES



**8.2 Data Cleaning and Validation :**

Steps Performed for indiviual files

1. Look for null values in each file
2. Replace Null values in indiviual file with following logic :

TimeStamp : bfill()

Categorical Data : Replace with a particular value for that column i.e Unknown

Numeric Data : Used Mean()

ID’s : Remove the entries with null Id’s

States/Country : Remove the entries with null values

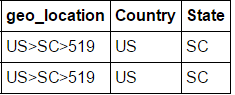
1. Platform value in Events file contained both Int and String value, String values were replaced with INT values:



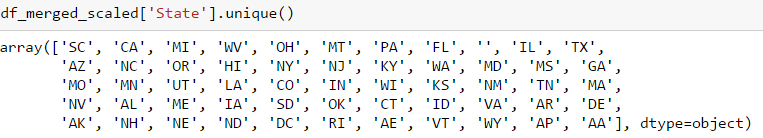
1. Geo location column in events, pages\_view\_sample file was divided into Country and State







1. All the unique values were checked in the Country and state column and null and erronous values were removed:

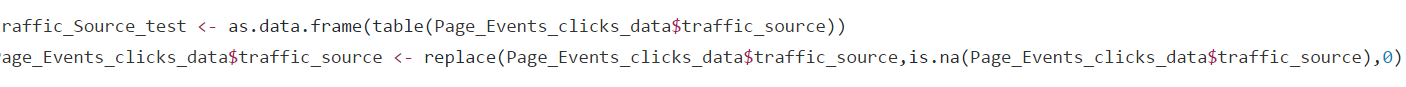








1. Null values in Traffic Source were replaced by 0



**8.3 METADATA MODELLING :**

OVERVIEW OF THE DATA FILES AND ITS RELATIONSHIP



STEPS TAKEN IN METADATA MODELLING

CHECK THE DATA SCHEMA

LIST OF DECREASING AD ID’S FOR EACH PAGE/DISPLAY ID

PROBABILITY OF INDIVIUAL AD GETTING CLICKED

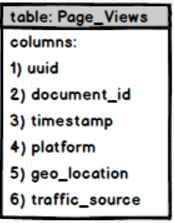
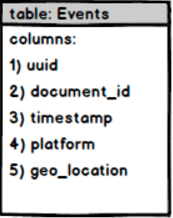
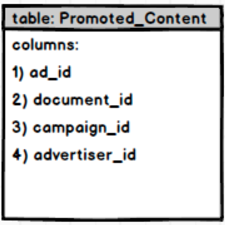
COMPARE SCHEMA BASED ON PROBLEM STATEMENT

CHECK FILES WHICH CAN BE JOINED

JOIN THE FILES TO CREATE SINGLE FILE

CHECK THE DATA SCHEMA

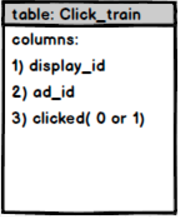
1. Each context (i.e. a set of recommendations) is given a display\_id. In each such set, the user has clicked on at least one recommendation. The identities of the clicked recommendations in the test set are not revealed.
2. Each user in the dataset is represented by a unique id (uuid). A person can view a document (document\_id), which is simply a web page with content (e.g.  a news article). On each document, a set of ads (ad\_id) are displayed. Each ad belongs to a campaign (campaign\_id) run by an advertiser (advertiser\_id).

INFORMATION ABOUT THE DISPLAY ID IN CONTEXT

INFORMATION ABOUT THE DOCUMENT THAT OPENS WHEN U CLICK AN AD  
  
NOTE: DOCUMENT\_ID DIFFERENT FROM DOCUMENT\_ID IN OTHER TABLES

INFORMATION ABOUT THE DOCUMENTS VIEWED BY THE USER AND HIS FEATURES

INFORMATION ABOUT THE DOCUMENT AS WELL AS CONFIDENCE ABOUT EACH DOC

SET OF ADS CLICKED OR NOT CLICKED FOR A DISPLAY ID

COMPARING SCHEMA AND PROBLEM STATEMENT

As the Document\_ID in Promoted Content is different from the Document\_ID a join is not possible between the events and promoted content files.

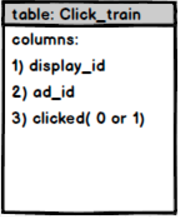
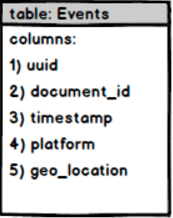
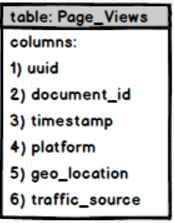
The promoted content file can only be joined if we are considering display\_id in our analysis.

Hence two different Combined files will be created for respective problem statement

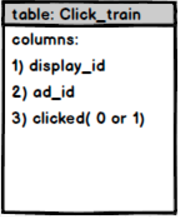
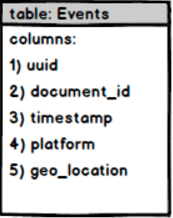
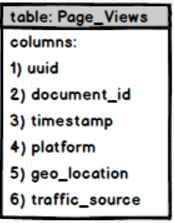
**For Problem Statement**

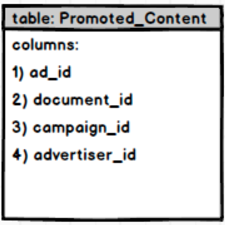
1. **Probablity of Indiviual AD getting** **clicked** : As Display\_ID is not needed in this scenario, Promoted Content will not be taken.

Following files will be joined to create a combined file for model creation:

1. **List of AD’s for a particular display in decreasing order :** As we are using display\_id in this scenario and not document\_id , we will be taking Promoted content and all the other files to create a single unified file, to take as many features as we can



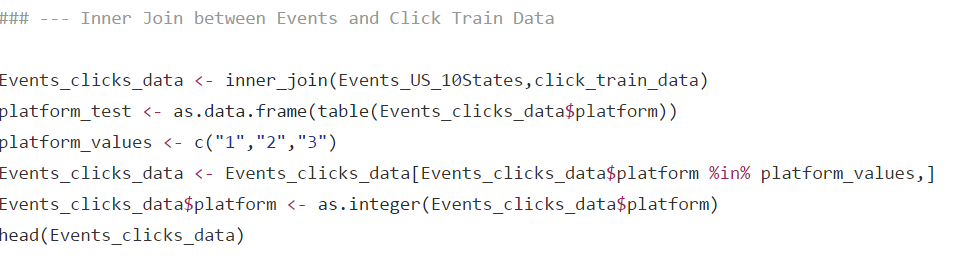
JOIN THE FILES TO CREATED MERGED FILES :

Different Approaches were taken to join the files for different problem statement.

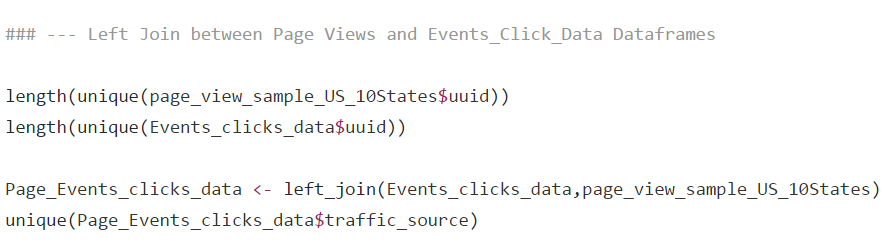
1. **Probablity of Indiviual AD getting** **clicked:**

Take the cleaned and validated files created above and join them as per the below specications.

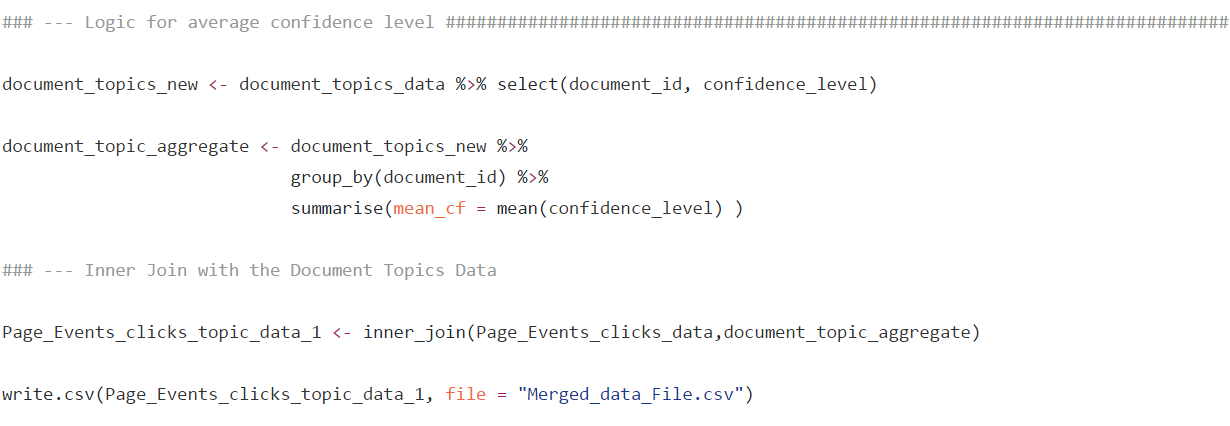
1. Inner join performed between events and click train data



1. Left Join as we need only Event\_click\_data and want to match with the same.

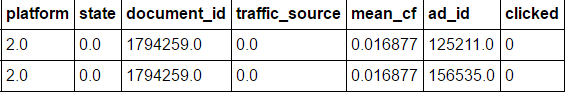


1. Calculate Mean Confidence level from the topics file for every document\_id
2. Join the above created file wth Document topic file to include mean confidence level



1. Merged\_data\_file is the final output file

MERGED FILE FOR PART ONE:



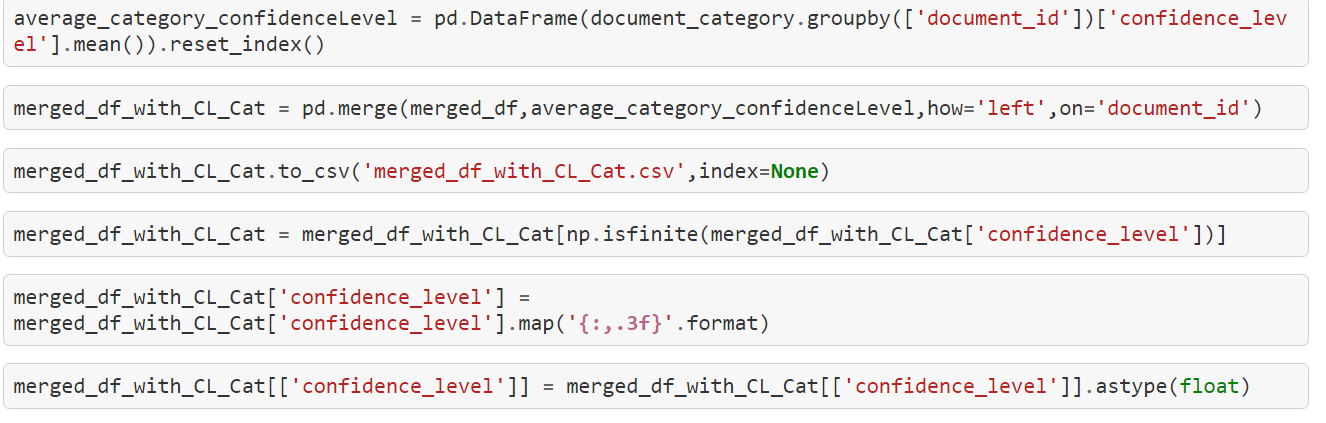
1. **List of AD’s for a particular display in decreasing order**
2. Left join between Events and click train data:



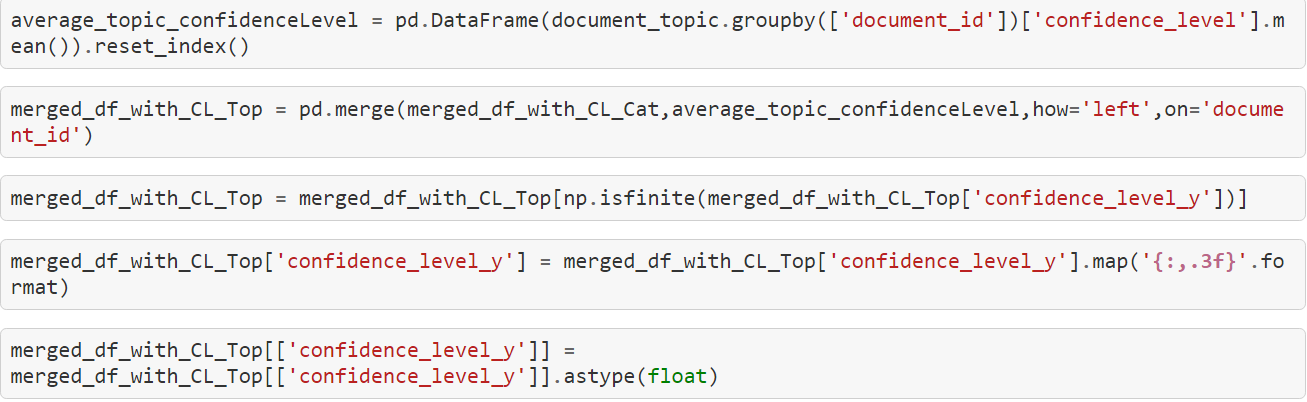
1. Left join between above created file and promoted file



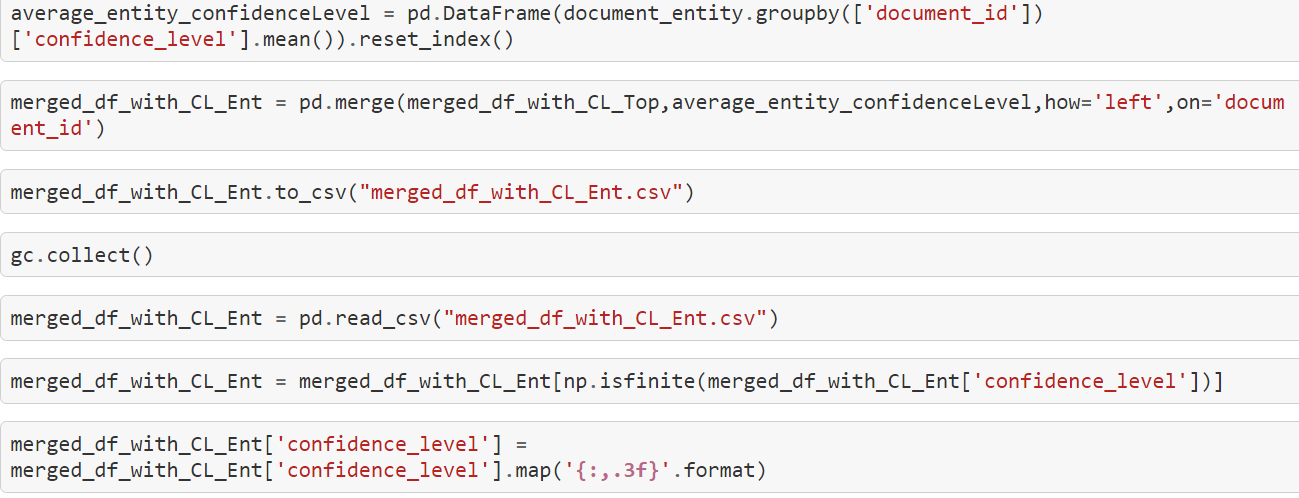
1. Left join with Document\_category file DERIVED COLUMN : AVERAGE CONFIDENCE LEVEL



1. Left join with Document\_topics file DERIVED COLUMN : AVERAGE CONFIDENCE LEVEL



1. Left join with Document\_entites file DERIVED COLUMN : AVERAGE CONFIDENCE LEVEL



MERGED FILE FOR PART TWO



1. **Indiviual Advertisement Probability**

Overview of the Flow

CHOSE APPROPRIATE FEATURES

LOAD FILE CREATED BY

SMOTE

RUN THE MODELS ON JUPYTER NOTEBOOK

BOOSTING

SVM

NEURAL

RANDOM FOREST

LOGISTIC

CHOSE THE BEST MODEL

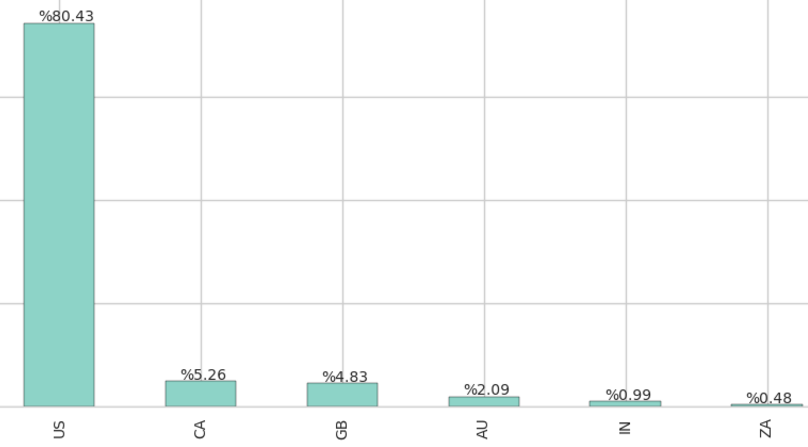
RUN MODEL ON AZURE

USE RSHINY TO CREATE WEB PAGE

CREATE WEB API

Two different scenarios were considered.   
One of them involved looking at the indiviual AD and calculating the probablity about that indiviual ad getting clicked.  
We will be exploring this in this section.

Looking at the data distribution for all the counteries , we infer that 80-81% of the data is concentrated for US. Hence, looking at US specific data makes more sense.



Only US country specific data contains States data which is one of the features being considered for the probability

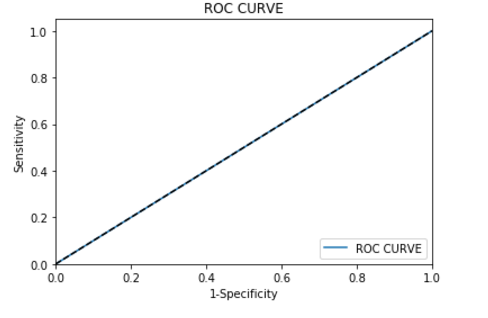
FEATURES CONSIDERED

Following features were considered for determing the probabilty of an indiviual AD

1. Platform
2. State
3. Traffic Source
4. Confidence Level
5. AD\_ID

OUTPUT: Probability of AD getting clicked

ROC CURVES GENERATED WITH FEATURES **(NO SMOTE)**



**THE CURVE SHOWS OVER SAMPLING AS VERY HIGH PRECISION AND LOW RECALL WAS GENERATED WHEN THE FEATURES WERE USED TO SCORE MODELS**

**APPLYING SMOTE** TO HANDLE OVER SAMPING

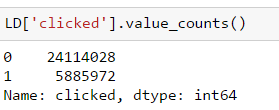
1. Looking at the data, we infer that the data is highly skewed in the direction of no clicks, i.e Clicked column has more than 85% data with value 0.
2. As only one AD will be clicked for every display id. Hence, the percentage of Clicked AD is considerably lower leading to over sampled data and very high precision.
3. SMOTE technique was used to solve this problem which balances the data for better attributes.

**SMOTE:**

SMOTE or Synthetic Minority Oversampling Technique is applied to an input dataset. This is a statistical technique for increasing the number of cases in your dataset in a balanced way.

You use SMOTE in datasets that are imbalanced. Typically, this means that the class you want to analyze is under-represented. There are many reasons for this: the category you are targeting might be very rare in the population, or the data might simply be difficult to collect. Regardless, SMOTE is a better way of increasing the number of rare cases than simply duplicating existing cases.

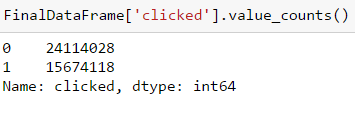
BEFORE SMOTE:



APPLYING SMOTE (RATIO :0.65) (TYPE: REGULAR SMOTE):



AFTER SMOTE:



1. 1’s and 0’s was now balanced in the ratio of 65% to 100% to reduce over sampling

CHOSING THE BEST MODEL IN JUPYTER NOTEBOOK

Models for which it was run

1. Logistic Regression
2. Random Forest
3. Neural Network

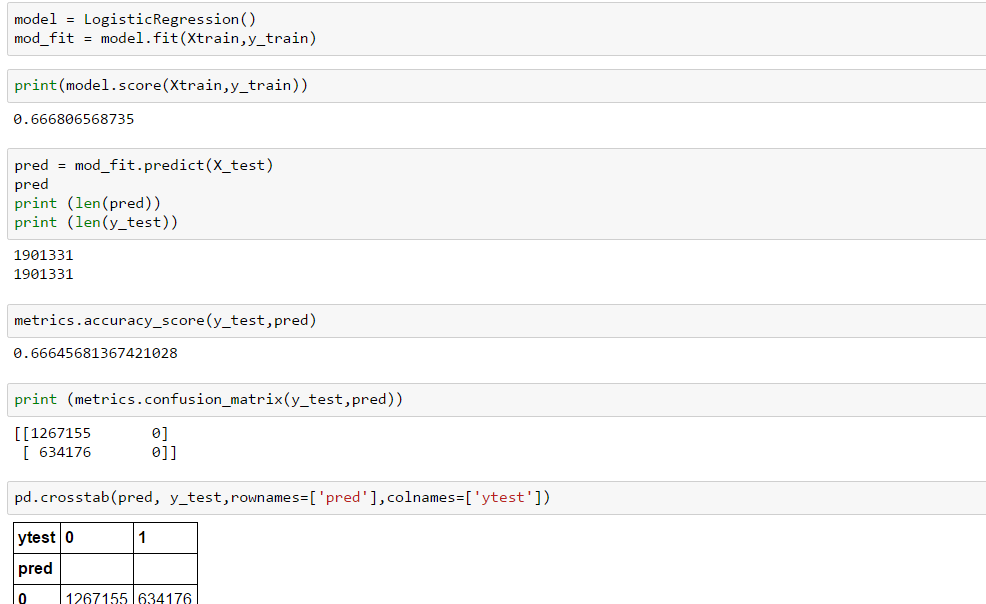
**Note: SVM and Boosted Algorithm were run directly on Azure due to memory issues in Python**

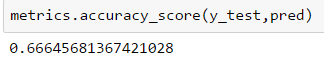
**Parameters:**

70% Data was taken for Training and 30% for Test

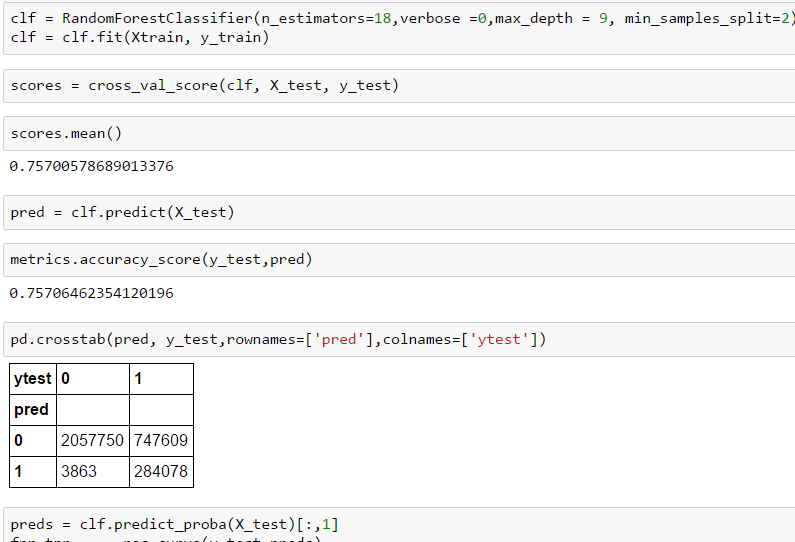
Different Parameters were used to tune the models (Screenshots below show the final parameters used)

**LOGISTIC REGRESSION:**



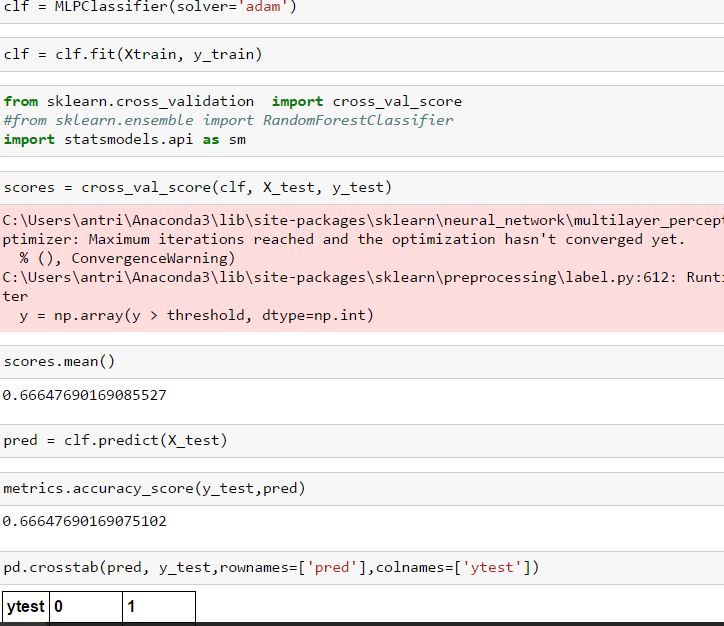


**RANDOM FOREST:**



**NOTE : DECREASING THE DEPTH OF THE TREE INCREASED THE ACCURACY AND PRECISION FOR US**

**NEURAL NETWORK:**

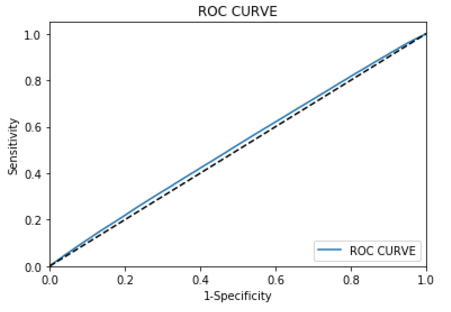
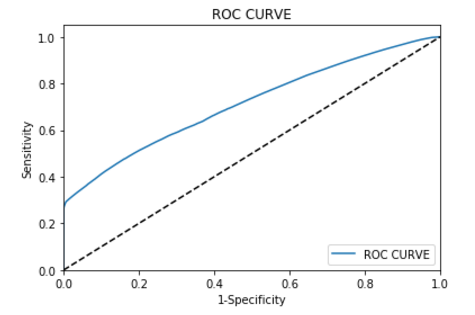


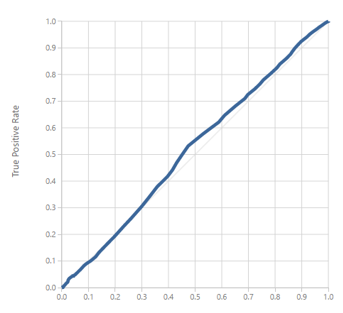
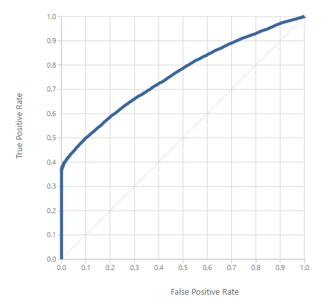
COMPARING ROC CURVES/ACCURACY FOR ALL THE MODELS

Note: SVM and Boosting Models were run directly on Azure (due to memory issues) :

LOGISTIC REGRESSION

RANDOM FOREST





SUPPORT VECTOR MACHINE

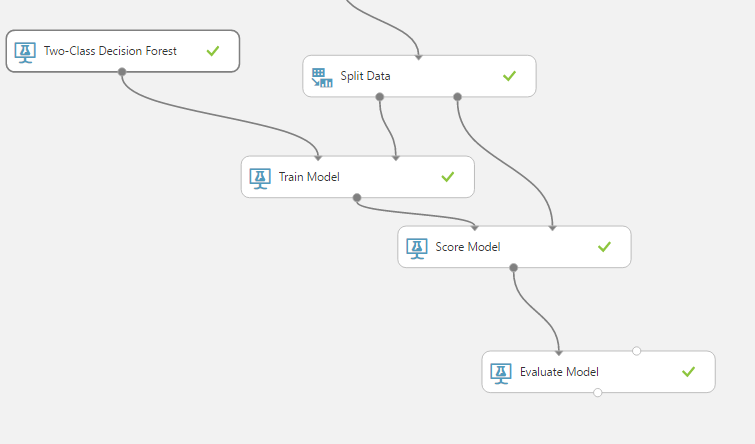
BOOSTING

|  |  |  |  |
| --- | --- | --- | --- |
| MODEL | **ACCURACY** | **PRECISION** | **RECALL** |
| Logistic Regression | 0.66 | 0.87 | 0.35 |
| Random Forest | 0.82 | 0.99 | 0.559 |
| SVM | 0.57 | 1 | 0.01 |
| Boosted Tree | 0.735 | 0.837 | 0.464 |
| Neural Network | 0.665 | 0.94 | 0.12 |

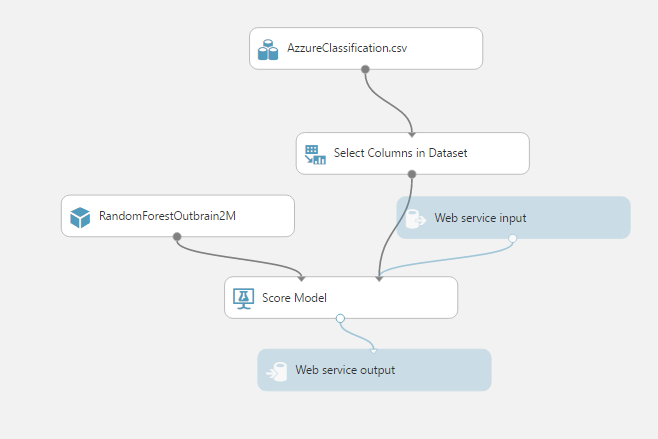
**MODEL CHOSEN: RANDOM FOREST (Random Forest had the best evaluation in our case).**

BEST MODEL IMPLEMENTATION ON AZURE (RANDOM FOREST)

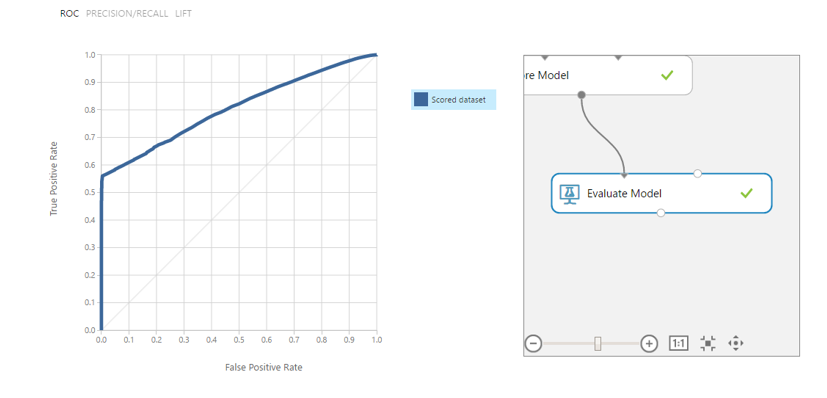
**Training Model :**



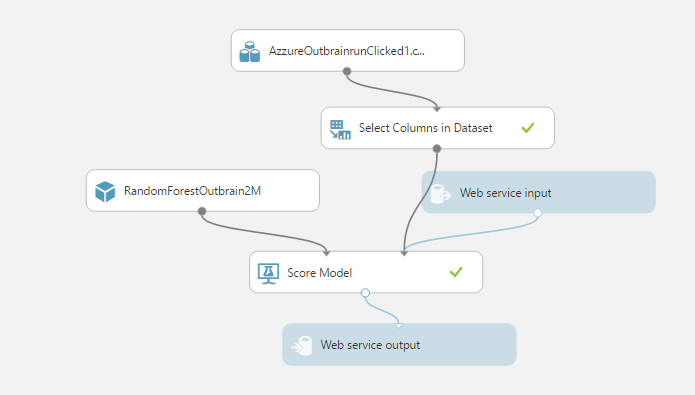
**Predictive Experiment (After saving the Trained Model)**



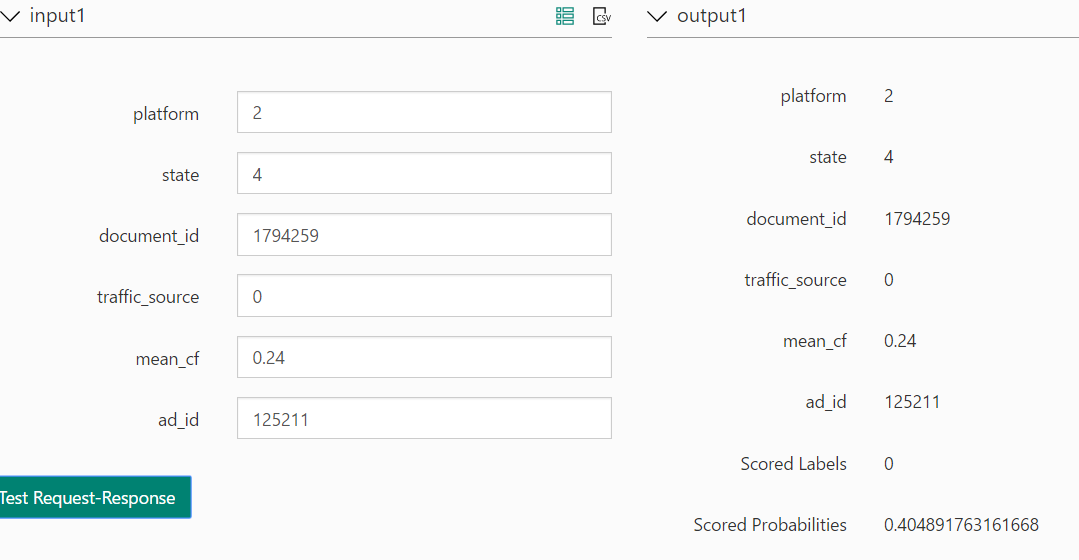
**EVALUATION FOR THE CHOSEN RANDOM FOREST ON AZURE:**



CREATING WEB SERVICE FOR THE MODEL

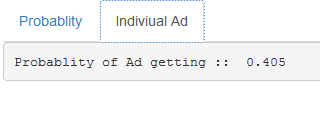
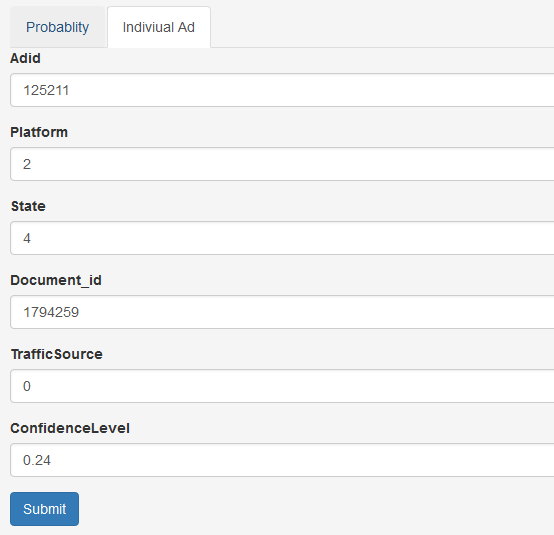


Screenshot of Input and Outut parameters when web API is tested on Azure:



IMPLEMENTATION ON RSHINY

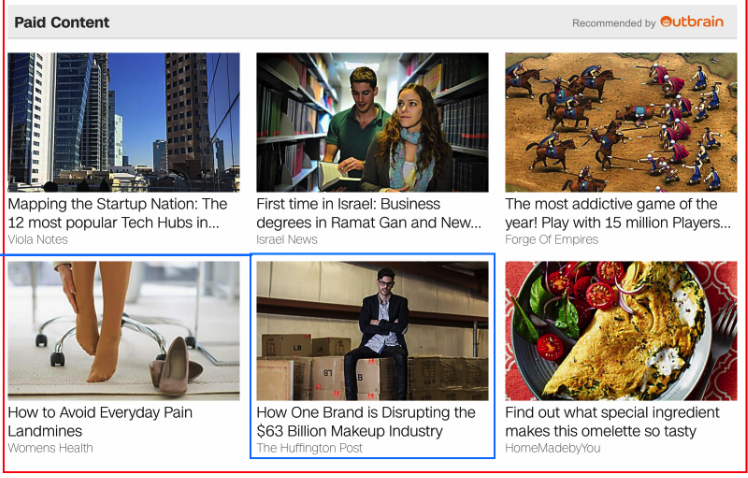
1. Enter the details on the R Shiny web page for a particular AD
2. The API created above in Azure will be called with the details entered
3. The probability of AD getting clicked will be displaced on the web page as shown below.



1. **List of AD’s for a particular display in decreasing order of click probability:**

In this section we will look at displaying all the AD id’s for a display ID in decreasing order of click probability

THIS USE CASE LOOKS LIKE BELOW



1. Here this particular page as shown above is One Display ID which will be entered by the user
2. All the Ad’s for that display will be shown but with decreasing order of click probabily

OVERVIEW OF THE FLOW

USE FILE CREATED FROM METADATA MODELLING

RUN THE MODELS ON JUPYTER NOTEBOOK

XGBOOST

LOGISTIC

RANDOM FOREST

CHOSE THE BEST MODEL

RUN MODEL ON AZURE

PYTHON SCRIPT FOR DECREASING PROBABILITY LIST

USE RSHINY TO CREATE WEB PAGE

CREATE WEB API

FEATURES CONSIDERED

1. Platform
2. Country
3. Traffic Source
4. Confidence Level (Three different for every document file)
5. Document\_id
6. Display\_id
7. Advertiser\_id
8. Campaign\_id

CHOSING THE CORRECT MODEL

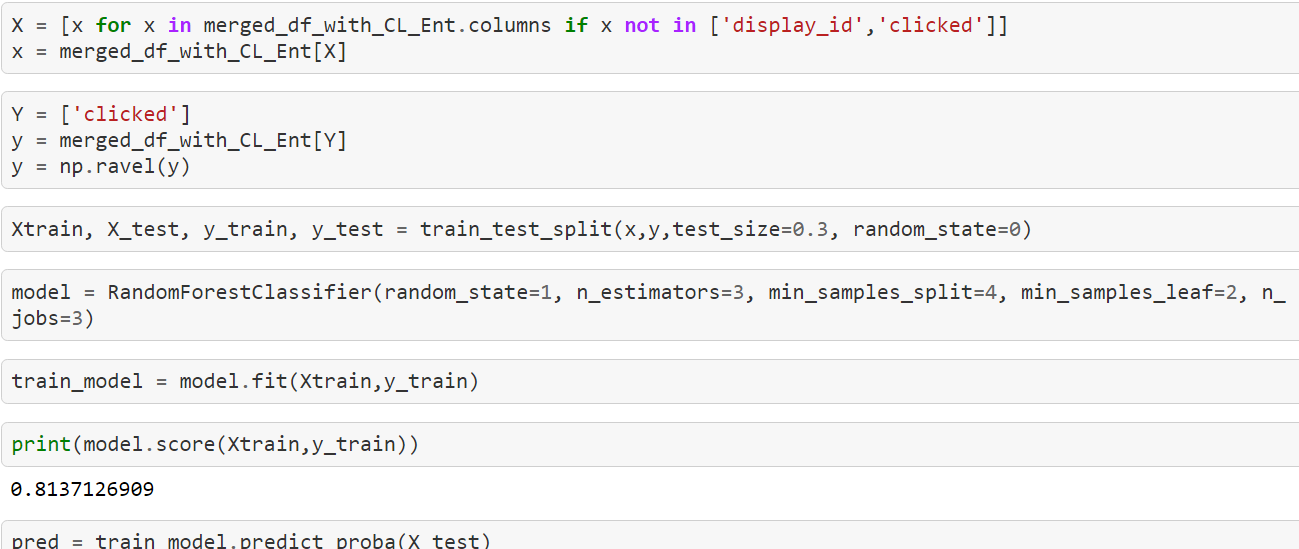
Three models were tried in jupyter notebook before the best model was used in Azure ML

1. Logistic Regression
2. Random Forest
3. XBoost
4. LOGISTIC REGRESSION

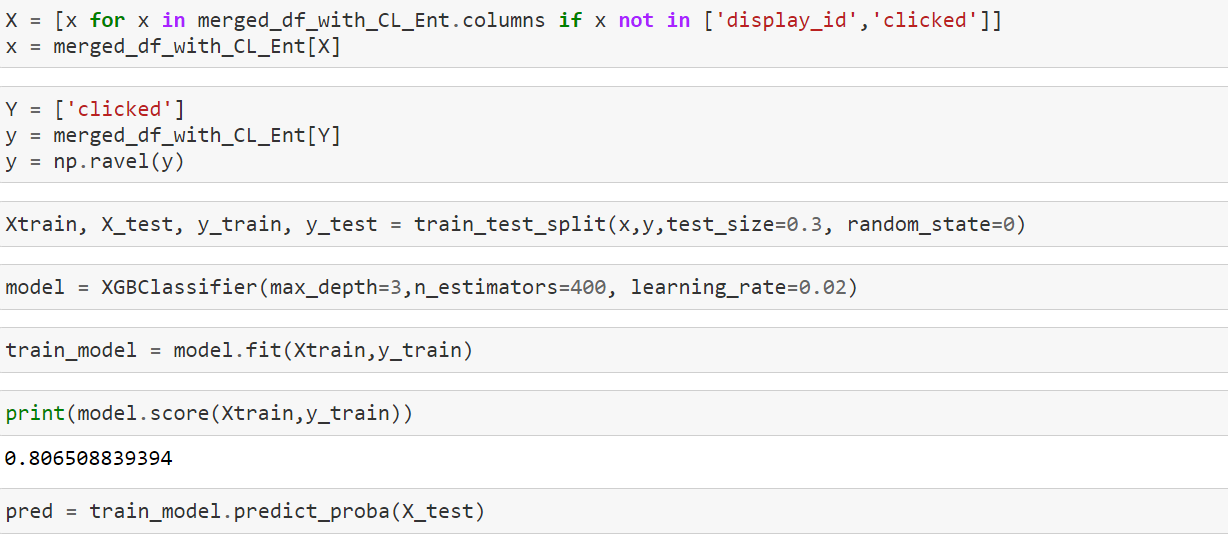




1. RANDOM FOREST



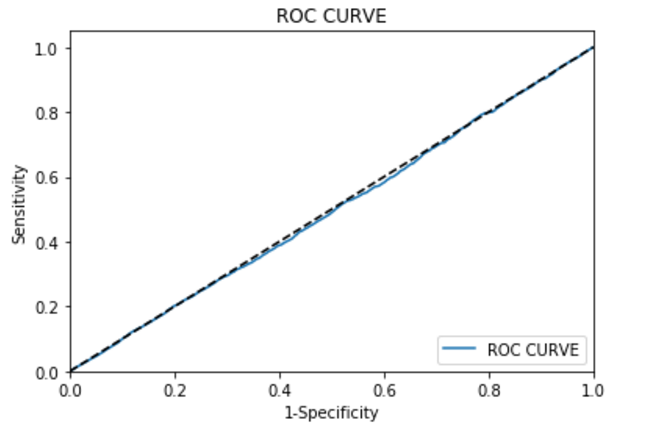
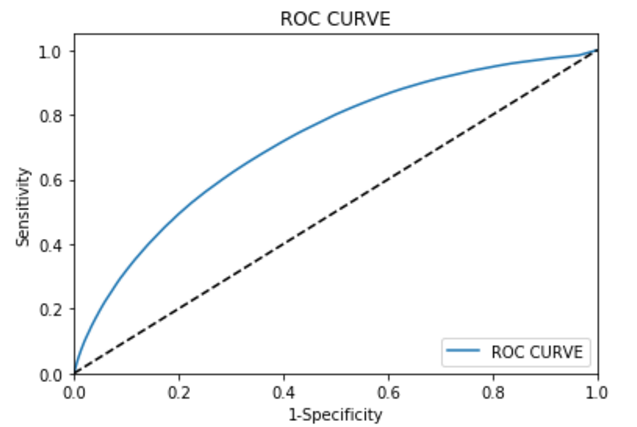
1. XGBOOST



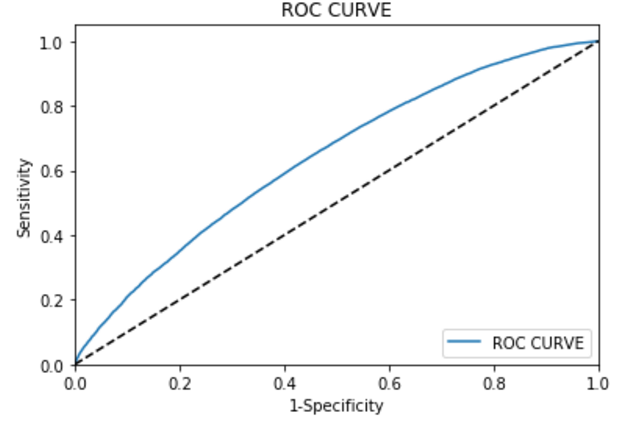
COMPARING ROC CURVES/ACCURACY FOR ALL THE MODELS

LOGISTIC REGRESSION

RANDOM FOREST



XGBOOST



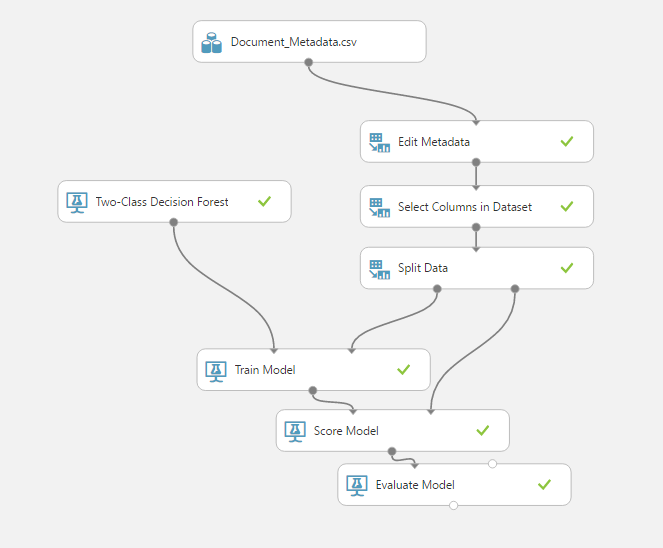
KAGGLE SUBMISSION FOR EACH MODEL AND SCORE (RANDOM FOREST HAD HIGHEST)



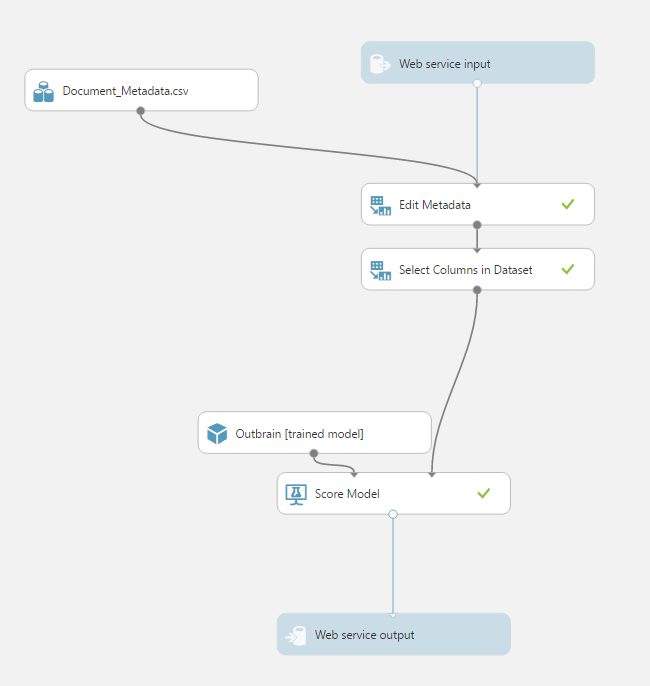
BEST MODEL CHOSEN: RANDOM FOREST

BEST MODEL IMPLEMENTATION ON AZURE (RANDOM FOREST)

**Training Model:**

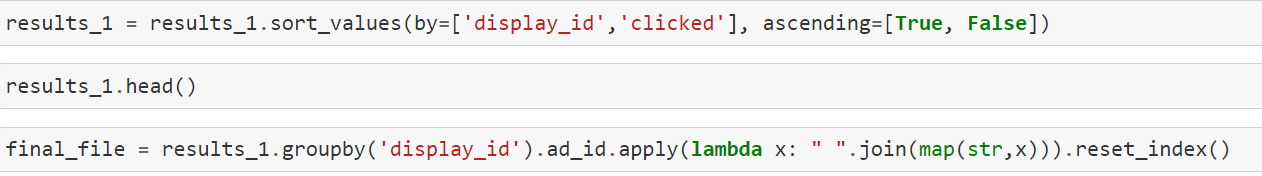


**Predictive Model**



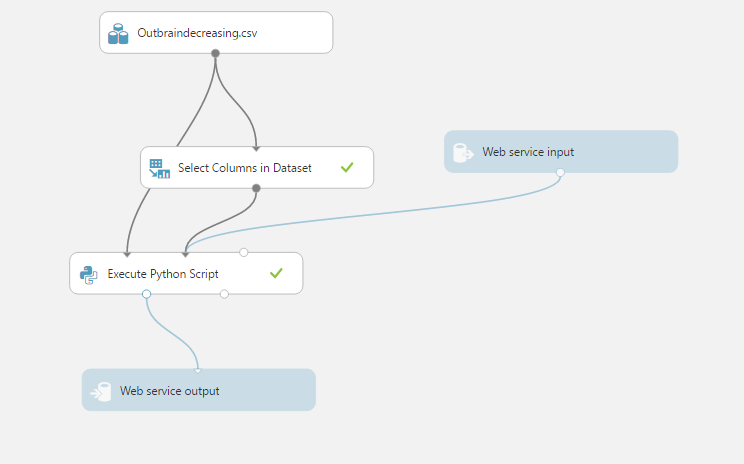
CREATING FILE WITH DECREASING AD PROBABILITY FOR EACH DISPLAY ID

The output of the above step was used to create the file with the decreasing AD Probability for Display ID

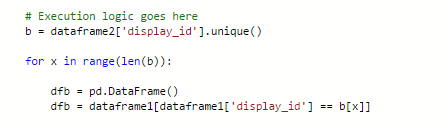


CREATING INPUT/OUTPUT FOR WEB API

Using the above file created. Created the following model:

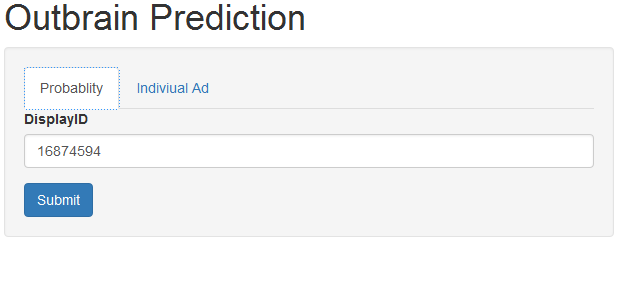


**Python Script:**

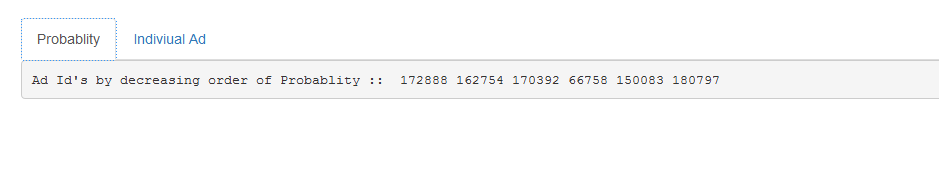


API DELOYED ON RSHINY

**INPUT:**



**OUTPUT ( AD Id’s in decreasing order of Probability):**



1. **REFERENCES**