Viewability in Programmatic Advertising

Charlene Chen

Agenda

- Background
- Hypothesis & Problem statement
- Dataset
- EDA
- Model
- Next Steps

Background

How we define if a video is viewable?

MRC VS. GroupM Standards

GroupM saw the MRC's guidelines as minimums by which we can trade. As a result, GroupM came up with stricter guidelines.

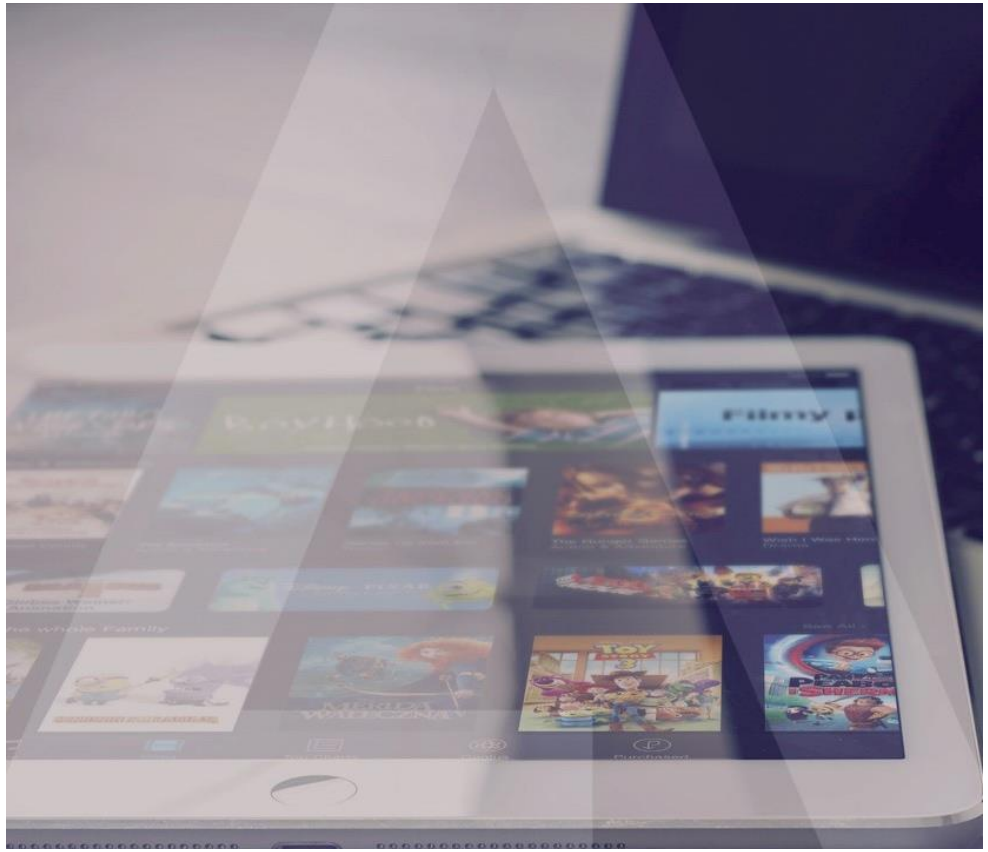
	Display	Video
MRC	50% of Ad for 1 Second	50% of Ad for 2 Consecutive Seconds
GroupM*	100% of Ad	100% of Ad for 50% of Video Duration Sound On/User Initiated

*assumes 100% NHT free

Viewability is here, it is real.

"An ad that is not seen by a human has zero value." - Ari Bluman

"At Xaxis, we believe strongly that viewability is a standard, not a KPI."



The Underlying Formula

Campaign	Impressions	View Measured Imps	Viewable Imps
Test	100,000	85,000	40,000

In-View Rate = Viewable Imps / Viewable Measured Imps
= 40,000/85,000 = 47.1%

*All data should come from the viewability vendors. No 3rd party ad server data should be used for any of the calculations.

Measurement: Viewability Vendors

DoubleVerify

Integral Ad Science

MOAT

ALENTY

Problem Statement

This project is to determine which features (advertiser vertical, domain/site, creative size, supply vendor, browser, non-human traffic rate, and small player size rate) will lead to higher (>=10.1%) GroupM IVR and what different roles they are playing (high coefficient vs. low coefficient) when it comes to higher GroupM IVR.

Hypothesis

Advertiser vertical, domain/site, creative size, supply vendor, browser, non-human traffic rate, and small player size rate will allow us to predict if a video will have high viewability. (GroupM IVR>=10.1%)

Dataset

GroupM: Use these measurements

MRC: Use these measurements

Search | Exports | Settings

Advertiser	Impressions Analyzed	Human Impressions	Human and Fully On-Screen Measurable Impressions	Human, Audible & Fully On-Screen for Half of Duration (15 sec. cap) Impressions	NHT %	In-View Measurable Impressions	2 Sec In-View Impressions	2 Sec Video In-View %
	3,756,051	3,465,983	3,356,065	1,117,974	7.72%	3,710,811	2,228,764	60.06%
jhcbia2	1,110,319	1,055,526	1,048,814	423,443	4.93%	1,104,898	834,992	75.57%
muzlyaz	594,545	565,524	563,427	250,658	4.88%	594,021	420,408	70.77%
oudr0mx	501,278	369,937	369,099	95,361	26.20%	500,939	257,321	51.37%
h23ak6v	360,959	356,345	343,258	172,728	1.28%	354,901	270,762	76.29%
ibekub67	307,992	291,147	241,833	20,796	5.47%	285,645	77,072	26.98%

January - March 2016 from MOAT

In [3]:
```
df=pd.read_csv("2016Q1_video_viewability_train.csv")
df
```

Out[3]:

	Moat Domain ID	Moat Domain Label	Size ID	Size Label	Supply Vendor ID	Supply Vendor Label	Advertiser ID	Advertiser Label	Browser	Impressions Analyzed	..
0	facebook.com	NaN	400x300	NaN	adaptv	NaN	rvqft21	Indeed.com	Chrome	926627	..
1	foodiewebsite.com	NaN	450x300	NaN	(unclassified)	NaN	qknuc79	Valeant	Internet Explorer	574514	..
2	facebook.com	NaN	400x300	NaN	(unclassified)	NaN	qknuc79	Valeant	Chrome	406049	..
3	thedoctorstv.com	NaN	600x450	NaN	(unclassified)	NaN	qknuc79	Valeant	Chrome	405838	..
4	facebook.com	NaN	400x300	NaN	(unclassified)	NaN	5b1h712	NaN	Chrome	318226	..
5	popcornflix.com	NaN	450x350	NaN	(unclassified)	NaN	h23ak6v	NBCU	Internet Explorer	304786	..

In [4]:
```
df.shape
```

Out[4]: (1782, 107)

EDA

- Missing value
- If the dataset is representative

```
In [9]: avgGroupM_IVR=df.GroupM_IVR.mean

In [10]: df.GroupM_IVR=df.GroupM_IVR.fillna(value=avgGroupM_IVR)

In [11]: df[df.GroupM_IVR.isnull()]
```

Out[11]:

	Moat Domain ID	Moat Domain Label	Size ID	Size Label	Supply Vendor ID	Supply Vendor Label	Advertiser ID	Advertiser Label	Browser	Impressions Analyzed	...	Hover Rate	Sum Time Until Hover (sec)	Time Until Hover (sec)	Small Player Sum	Small Player Rate	Below the Fold Sum	Be the Fo Ra

0 rows × 107 columns

Programmatic Video Desktop Benchmarks | Q1 2016 MOAT

SUMMARY BENCHMARKS

99.2% In-View Measurable %	59.8% On-Screen %	51.7% 1 Sec Video In-View %	47.7% 2 Sec Video In-View %	46.2% Human and Viewable %
46.3% 3 Sec Video In-View %	44.0% 5 Sec Video In-View %	42.3% Fully On-Screen % (No Time Minimum)	36.0% 1 Sec Fully On-Screen %	33.2% 3 Sec Fully On-Screen %
11.5% Audible and Fully On-Screen for Half of Duration %	10.1% Human, Audible & Fully On-Screen for Half of Duration (15 sec. cap) %	16.2% Audible and 80% On-Screen for Half of Duration (15 sec. cap) %	18.6s In-View Time	81.4% % of Video Played In-View
64.3% Reached Complete %	12.1% Audible and Visible on Complete %	22.2% Completion Quality	15.6% Out of Focus %	31.7% Hostile IFrame %

GroupM IVR

```
High_IVR_df = df[df.GroupM_IVR >= 10.1]
```

```
High_IVR_df.shape
```

```
(746, 107)
```

```
746/1782.0
```

```
0.4186307519640853
```

###This dataset includes 41.9% videos which have high GroupM viewability

```
df.GroupM_IVR.mean()
```

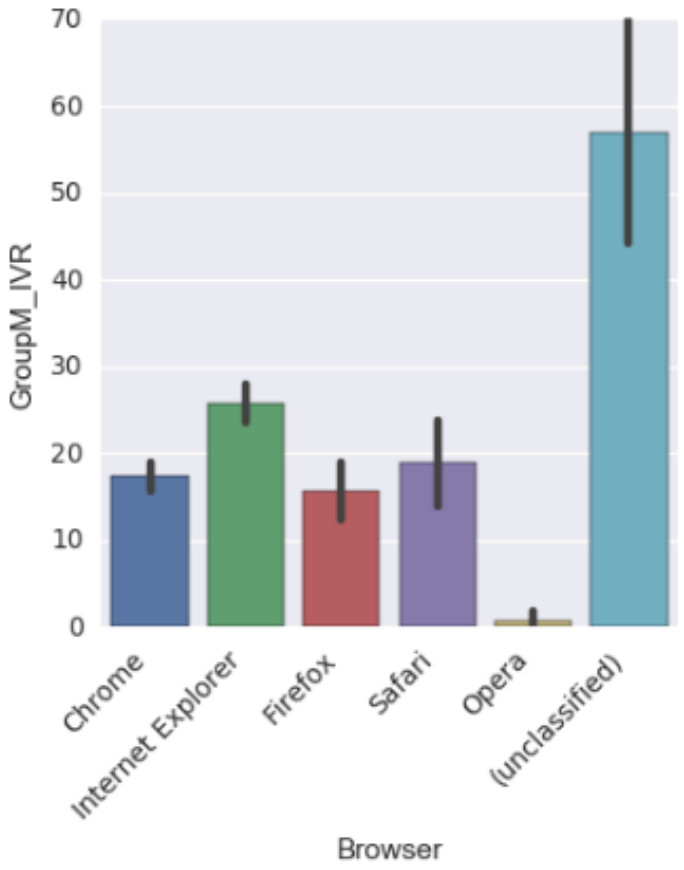
```
19.96594048287478
```

This dataset is a good sample!

```
In [253]: df.groupby('Browser')['binary'].value_counts(inplace=True)
```

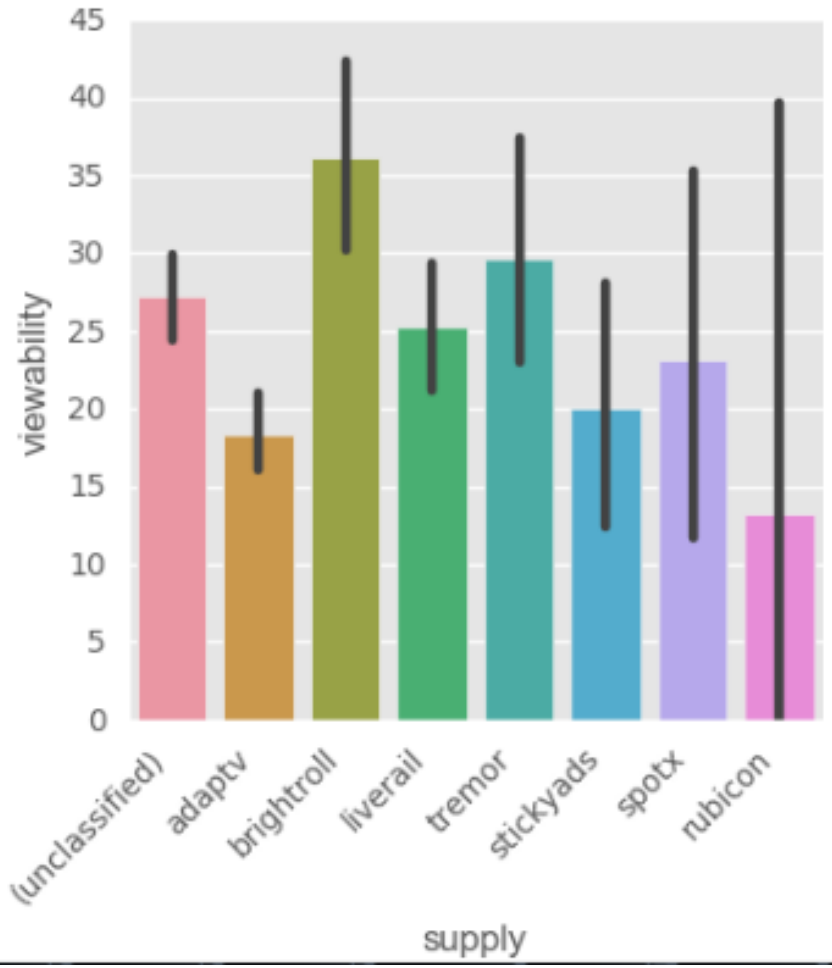
```
Out[253]: Browser            binary
          (unclassified)     1           2
          Chrome             0         636
                             1         262
          Firefox            0         145
                             1          52
          Internet Explorer  1         295
                             0         292
          Opera              0          10
          Safari             0          61
                             1          27
          Name: binary, dtype: int64
```

```
Out[256]: <seaborn.axisgrid.FacetGrid at 0x1eabcb70>
```



```
p = sb.factorplot(x='supply',
                  y='viewability',
                  kind='bar',
                  data = X,
                  size = 4
                )
p.set_xticklabels(rotation=45, horizontalalignment='right')
```

```
Out[255]: <seaborn.axisgrid.FacetGrid at 0x11a7a860>
```



#get_dummies to convert categorical features into binary features

In [262]:
```
def describe_categorical(X):
    from IPython.display import display, HTML

    display(HTML(X[X.columns[X.dtypes=='object']].describe().to_html()))
```

In [263]:
```
describe_categorical(X)
```

	advertiser	supply	smlplayer
count	1035	1035	1035
unique	13	8	2
top	Revlon	(unclassified)	0.00%
freq	223	417	935

In [264]:
```
# use get_dummies to convert categoricals into binary features
categorical_variables=['advertiser','supply','smlplayer']

for variable in categorical_variables:
#create array of dummies
    dummies = pd.get_dummies(X[variable], prefix=variable)
#update X to include dummies and drop main variable
    X=pd.concat([X, dummies],axis=1)
    X.drop([variable], axis =1, inplace=True)
```

Model: Decision Trees

In [268]:
```
###Build a decision tree model to predict the "high/low viewability" of a given video
from sklearn.tree import DecisionTreeClassifier

y=X['Binary']

X.drop('Binary', axis=1, inplace=True)
model = DecisionTreeClassifier()
```

In [269]:
```
# Fits the model
model.fit(X, y)
```

Out[269]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

In [286]:
```
###Evaluate the decision tree using cross-validation; use AUC as the evaluation metric.
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X, y, scoring='roc_auc', cv=5)
print('CV AUC {}, Average AUC {}'.format(scores, scores.mean()))
```

```
CV AUC [ 0.94880022  0.92666104  0.90779842  0.91380443  0.89184448], Average AUC 0.917781717632
```

Random Forest Classifier

In [288]:
```
###Build a random forest model to predict the the viewability of a video.
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators =100)

model.fit(X, y)
```

Out[288]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=100, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

Evaluate the Random Forest model using cross-validation; increase the number of estimators and view how that improves predictive performance.

```
In [290]: scores = cross_val_score(model, X, y, scoring='accuracy')
          print('CV Accuracy {}, Average Accuracy {}'.format(scores, scores.mean()))

          for n_trees in range(1, 150, 10):
              model = RandomForestClassifier(n_estimators = n_trees)
              scores = cross_val_score(model, X, y, scoring='accuracy')
              print('n trees: {}, CV accuracy {}, Average accuracy {}'.format(n_trees, scores, scores.mean()))

CV Accuracy [ 0.8583815   0.8173913   0.82267442], Average Accuracy 0.832815741948
n trees: 1, CV accuracy [ 0.8150289   0.75362319  0.75872093], Average accuracy 0.775791006791
n trees: 11, CV accuracy [ 0.84393064  0.8115942   0.79651163], Average accuracy 0.817345488881
n trees: 21, CV accuracy [ 0.86705202  0.8         0.81395349], Average accuracy 0.827001837164
n trees: 31, CV accuracy [ 0.8583815   0.8173913   0.82848837], Average accuracy 0.834753726444
n trees: 41, CV accuracy [ 0.85549133  0.80289855  0.81976744], Average accuracy 0.826052440688
n trees: 51, CV accuracy [ 0.84393064  0.8173913   0.82267442], Average accuracy 0.827998786264
n trees: 61, CV accuracy [ 0.86705202  0.82608696  0.8255814 ], Average accuracy 0.839573458331
n trees: 71, CV accuracy [ 0.84682081  0.83188406  0.81104651], Average accuracy 0.829917126282
n trees: 81, CV accuracy [ 0.84971098  0.81449275  0.81686047], Average accuracy 0.827021400466
n trees: 91, CV accuracy [ 0.86416185  0.82028986  0.81104651], Average accuracy 0.831832738804
n trees: 101, CV accuracy [ 0.85549133  0.82318841  0.82267442], Average accuracy 0.833784717961
n trees: 111, CV accuracy [ 0.85260116  0.8173913   0.82267442], Average accuracy 0.830888959674
n trees: 121, CV accuracy [ 0.84971098  0.82608696  0.80813953], Average accuracy 0.827979158021
n trees: 131, CV accuracy [ 0.86416185  0.82028986  0.81976744], Average accuracy 0.834739715548
n trees: 141, CV accuracy [ 0.85260116  0.83478261  0.81976744], Average accuracy 0.835717068875
```

Check Feature Importance

	Features	Importance Score
0	hover	0.445989
1	btf	0.177846
4	Internet Explorer	0.051982
11	advertiser_Indeed.com	0.044253
2	Chrome	0.030037
28	smlplayer_0.00%	0.022486
29	smlplayer_100.00%	0.021397
8	advertiser_Church and Dwight	0.019005
15	advertiser_Revlon	0.014390
20	supply_(unclassified)	0.014233
21	supply_adaptv	0.013980
22	supply_brightroll	0.013331
18	advertiser_Valeant	0.013065
23	supply_liverail	0.012457
12	advertiser_NBCU	0.012412

X.corr( )

	hover	btf	Binary	Chrome	Firefox	Internet Explorer
hover	1.000000	-0.318352	0.47904	-0.274742	-0.100049	0.277401
btf	-0.31835	1.000000	-0.294508	0.130648	0.03107	-0.148275
Binary	0.47904	-0.294508	1.000000	-0.251744	-0.044767	0.30794
Chrome	-0.27474	0.130648	-0.251744	1.000000	-0.282472	-0.76779
Firefox	-0.10005	0.03107	-0.044767	-0.282472	1.000000	-0.221539
Internet Explorer	0.277401	-0.148275	0.30794	-0.76779	-0.221539	1.000000
Safari	0.150328	0.00828	-0.046452	-0.222921	-0.064322	-0.174834
advertiser_Blackrock	0.125874	-0.087175	0.137845	-0.0824	-0.014264	0.081399
advertiser_Church and Dwight	0.132419	-0.058738	0.172489	0.128957	-0.037272	-0.090699
advertiser_IKEA	0.05596	0.030442	0.096693	-0.057392	-0.043039	0.099543
advertiser_Indeed.com	-0.13665	-0.047675	-0.301627	0.158122	-0.028266	-0.164672
advertiser_NBCU	-0.03323	0.216426	0.02428	-0.041903	0.058983	-0.009463
advertiser_Revlon	0.068397	-0.11764	0.116862	-0.16599	0.010631	0.141612
advertiser_Valeant	-0.10197	0.120005	-0.117002	0.061059	0.011919	-0.03897
supply_(unclassified)	-0.02598	0.13204	0.051405	-0.028699	0.041594	0.033645
supply_adaptv	0.107456	-0.138008	-0.078212	0.041419	-0.071755	-0.027046
supply_brightroll	-0.0565	0.069936	0.09529	0.002992	0.031887	-0.019611
supply_liverail	0.029054	-0.056407	-0.009298	-0.1048	0.010935	0.067907
supply_tremor	-0.05031	-0.10722	-0.016502	0.070045	-0.018741	-0.043208
smlplayer_0.00%	0.229509	-0.426781	0.264193	-0.003476	-0.042912	-0.002805
smlplayer_100.00%	-0.22951	0.426781	-0.264193	0.003476	0.042912	0.002805

Using Logistic Regression model to compare accuracy score

In [455]:
```
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
Scaler =StandardScaler()
X=Scaler.fit_transform(X)
```

In [456]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=42)
```

In [457]:
```
# TODO
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression(C=10**2)
#X=X_train[df.columns[1:]]
#y=y_train['Binary']
output=LR.fit(X_train,y_train)
```

In [458]:
```
preds = output.predict(X_test)
```

In [459]:
```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_test,preds)
print "LR accuracy is {:.2f}".format(score)
```

```
LR accuracy is 0.84
```

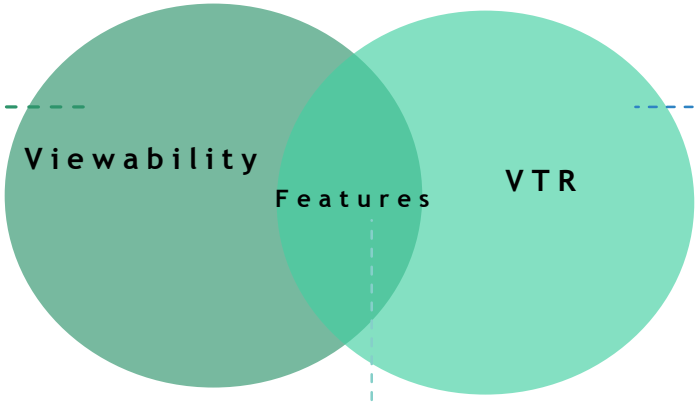
Next Steps

“ Viewability is not about ad effectiveness nor ad engagement. It is simply the delivery of ads that render on the screen. In other words, the opportunity to be seen. ”

- IAB

Viewability VS. View Through Rate

Determines whether or not the video ads can be seen



Determines whether the viewers are engaging with the ad – a most frequently used KPI for video campaigns

Commonly shared attributes and signals lead to both high Viewability and high VTR

*VTR = Complete views (user did not skip) / Impressions (ad rendered)

Thank You!