Machine Learning Project Phase 1

Predicting Revenue-Related Metrics using Google ads data

Name: Minh Phan

Student number: s3335814

Predicting Revenue-Related Metrics using Google ads data

April 26, 2019

1 Introduction

The objective of this project is to predict the revenue-related metric using the dataset provided by the client via the Kaggle competition. This project has two phases. Phase I focuses on data preprocessing and exploration, as covered in this report. We shall present model building in Phase II. The rest of this report is organised as follows. Section 1 describes the data sets and their attributes. Section 2 covers data pre-processing. In Section 3, we explore each attribute and their inter-relationships. The last section presents a brief summary. Compiled from Jupyter Notebook, this report contains both narratives and the Python codes used for data pre-processing and exploration.

1.1 Data Sets

This dataset contains online advertising data where the target feature is a revenue-related metric and the descriptive features are various advertising metrics and characteristics. Each row represents a website traffic record that comes from a specific country, company, and device type combination. The dataset contains 30 days of training data and 5 days of test data. The training data contains about 215K records and the test data contains about 31K records.

1.1.1 Target Feature

The target feature is a revenue-related metric. The variable is continous ranging from 0.000098 to 47.060000 in the traing data. The metric measures the website traffic.

1.1.2 Descriptive Features

The variable description is produced here from `Advertising_Data_Description.pdf` file:

- `companyId`: Company ID of record (categorical)
- `countryId`: Country ID of record (categorical)
- `deviceType`: Device type of record (categorical corresponding to desktop, mobile, tablet)
- `day`: Day of record (integer between 1 (oldest) and 30 for train, 31 and 35 (most recent) for test)
- `dow`: Day of week of the record (categorical)

- `price1, price2, price3`: Price combination for the record set by the company (numeric)
- `ad_area`: area of advertisement (normalized between 0 and 1)
- `ad_ratio`: ratio of advertisement's length to its width (normalized between 0 and 1)
- `requests, impression, cpc, ctr, viewability`: Various metrics related to the record (numeric)
- `ratio1, ..., ratio5`: ratio characteristics related to the record (normalized between 0 and 1)
- `y (target feature)`: revenue-related metric (numeric)

  Google ads metric explains:

requests: is counted whenever your site requests ads to be displayed.
impression: how often an ad is shown on Google and Google network.
cpc: cost per click.
ctr: click through rate, = clicks/ impressions.
viewability: portion of the ad that is seen be a user at the time.

2 Data Pro-processsing

2.1 Preliminaries

We read the training and test datasets from the file advertising_train.csv.

```
In [1]: import numpy as np
```

```
In [2]: import pandas as pd
        import warnings
        warnings.filterwarnings('ignore')
```

```
In [3]: ad=pd.read_csv("advertising_train.csv")
```

2.2 Data Cleaning and Transformation

```
In [4]: print(f"Dimension of the data set is {ad.shape} \n")
        print(f"Data Types are: ")
        print(ad.dtypes)
```

```
Dimension of the data set is (214128, 21)

Data Types are:
companyId        int64
countryId        int64
deviceType       int64
day            float64
dow             object
price1         float64
```

```
price2          float64
price3          float64
ad_area         float64
ad_ratio        float64
requests        float64
impression      float64
cpc             float64
ctr             float64
viewability     float64
ratio1          float64
ratio2          float64
ratio3          float64
ratio4          float64
ratio5          float64
y               float64
dtype: object
```

Firstly, we need to make sure the data types of the features are appropriate.
For categorical features:

```
In [5]: categoricalColumn = ['companyId', 'countryId', 'deviceType', 'dow']
        for col in categoricalColumn: ad[col] = ad[col].astype('category')
```

We can remove the day variable since it doesnt have any predicting values.

```
In [6]: ad = ad.drop('day', axis=1)
```

```
In [7]: ad['impression'] = ad['impression'].astype('int64')
```

```
In [8]: print(ad.dtypes)
```

```
companyId       category
countryId       category
deviceType      category
dow             category
price1           float64
price2           float64
price3           float64
ad_area          float64
ad_ratio         float64
requests         float64
impression         int64
cpc              float64
ctr              float64
viewability      float64
ratio1           float64
ratio2           float64
ratio3           float64
```

```
ratio4          float64
ratio5          float64
y               float64
dtype: object
```

On surface, no attributes contain NaN values (though the missing values might be coded with different labels) as shown in the code chunk.

```
In [9]: print(f"\nNumber of missing value for each feature:")
        print(ad.isnull().sum())
```

```
Number of missing value for each feature:
companyId      0
countryId      0
deviceType     0
dow            0
price1         0
price2         0
price3         0
ad_area        0
ad_ratio       0
requests       0
impression     0
cpc            0
ctr            0
viewability    0
ratio1         0
ratio2         0
ratio3         0
ratio4         0
ratio5         0
y              0
dtype: int64
```

From Table 1 and 2 , we can observe a few problems.

Firstly, for ad_ration,ad_area, ratio2, ratio3, ratio4 and ratio5, the values are meant to be normalised from 0 to 1. However, we can see that the max values of these values are larger then 1.

Secondly, ctr(click through) represents percentage , however we can see its maximum value is 2.00.

Lastly, the deviceType variables only has 3 options: mobile, tablet, desktop. However we observed 4 options in table 3

```
In [10]: from IPython.display import display, HTML
         display(HTML('<b>Table 1: Summary of continuous features</b>'))
         display(ad.describe(include ='float64'))
```

```
    display(ad.describe(include ='int64'))

    display(HTML('<b>Table 2: Summary of categorical (object) features</b>'))
    display(ad.describe(include = 'category'))
```

```
<IPython.core.display.HTML object>


              price1         price2         price3        ad_area  \
count  214128.000000  214128.000000  214128.000000  214128.000000
mean        0.438229       0.630178       0.932436       4.724445
std         1.281403       1.481552       1.839991       6.273410
min         0.000000       0.000000       0.000000       0.000100
25%         0.000000       0.000000       0.000000       0.000100
50%         0.010000       0.090000       0.294800       0.000100
75%         0.190000       0.570000       0.985650       7.500000
max        14.690000      63.120000      78.900000      36.000000

            ad_ratio      requests            cpc            ctr  \
count  214128.000000  2.141280e+05  214128.000000  214128.000000
mean        0.923402  8.678997e+03       0.177862       0.032921
std         0.482055  1.223472e+05       0.707260       0.092502
min         0.083330  0.000000e+00       0.000000       0.000000
25%         0.833330  0.000000e+00       0.000000       0.000000
50%         1.000000  1.470000e+02       0.015700       0.001700
75%         1.000000  1.633000e+03       0.125200       0.012000
max         5.000000  6.701924e+06     132.533900       2.000000

         viewability         ratio1         ratio2         ratio3  \
count  214128.000000  214128.000000  214128.000000  214128.000000
mean        0.377929       0.558284       0.491079       0.311646
std         0.365938       0.446955       0.414312       0.444088
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.331500       0.750000       0.627100       0.027600
75%         0.715900       1.000000       0.895700       1.000000
max         7.000000       1.000000       1.027000       1.500000

              ratio4         ratio5              y
count  214128.000000  214128.000000  214128.000000
mean        0.131008       0.188300       0.847004
std         0.239758       0.297121       1.390593
min         0.000000       0.000000       0.000098
25%         0.000000       0.000000       0.150415
50%         0.000000       0.000000       0.419000
75%         0.163600       0.384700       0.959048
max         1.076900       1.200000      47.060000
```

```
         impression
count  2.141280e+05
mean   5.585714e+03
std    9.871334e+04
min    0.000000e+00
25%    0.000000e+00
50%    9.900000e+01
75%    1.058000e+03
max    6.100324e+06
```

```
<IPython.core.display.HTML object>
```

```
        companyId  countryId  deviceType       dow
count      214128     214128      214128    214128
unique          6        163           4         7
top            43        234           2  Saturday
freq       134655      21507       94827     35200
```

2.2.1 Continuous Features

As discussed in previous section, we need to investigate the ad_ratio,ad_area, ratio2, ratio3, ratio4 and ratio5 variables.

For the ad_ratio, ratio2, ratio3, ratio4 and ratio5 variables, majority of the observations have values less than 1, therefore we can treat ones with ratio lager than 1 as possible mistakes and remove them from the data sets.

```
In [11]: ad_removed = ad[ad.ratio2 <= 1]

In [12]: ad_removed = ad_removed[ad_removed.ratio3 <= 1]

In [13]: ad_removed = ad_removed[ad_removed.ratio4 <= 1]

In [14]: ad_removed = ad_removed[ad_removed.ratio5 <= 1]

In [15]: ad_removed = ad_removed[ad_removed.ad_ratio <= 1]

In [16]: ad_removed.describe()
```

```
Out[16]:               price1         price2         price3        ad_area  \
         count  187561.000000  187561.000000  187561.000000  187561.000000
         mean        0.454444       0.631557       0.918268       3.672070
         std         1.312231       1.477399       1.806672       5.231608
         min         0.000000       0.000000       0.000000       0.000100
         25%         0.000000       0.000000       0.000000       0.000100
         50%         0.010000       0.080000       0.283700       0.000100
         75%         0.190000       0.570000       0.957300       7.500000
```

```
max        14.690000      29.540000      29.544300      36.000000

              ad_ratio       requests     impression            cpc  \
count  187561.000000  1.875610e+05  1.875610e+05  187561.000000
mean        0.824727  7.500860e+03  5.012452e+03       0.141920
std         0.292997  1.237876e+05  1.012706e+05       0.501185
min         0.083330  0.000000e+00  0.000000e+00       0.000000
25%         0.833330  0.000000e+00  0.000000e+00       0.000000
50%         1.000000  9.600000e+01  6.400000e+01       0.011700
75%         1.000000  1.202000e+03  7.860000e+02       0.101300
max         1.000000  6.701924e+06  6.100324e+06      50.938900

                   ctr    viewability         ratio1         ratio2  \
count  187561.000000  187561.000000  187561.000000  187561.000000
mean        0.035379       0.379390       0.544189       0.479616
std         0.095907       0.372985       0.451954       0.418858
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.001700       0.331700       0.733800       0.602500
75%         0.015300       0.727600       1.000000       0.897000
max         1.000000       7.000000       1.000000       1.000000

                ratio3         ratio4         ratio5              y
count  187561.000000  187561.000000  187561.000000  187561.000000
mean        0.272145       0.136655       0.203209       0.870772
std         0.422794       0.240495       0.304442       1.445689
min         0.000000       0.000000       0.000000       0.000098
25%         0.000000       0.000000       0.000000       0.147222
50%         0.015300       0.000000       0.000000       0.417857
75%         0.572600       0.189500       0.440800       0.979310
max         1.000000       1.000000       1.000000      47.060000
```

After the removals of those observations we still have majority of the data left

With the variable ad_area, almost half of the value is larger than 1. This could potentially be the result of the miss interpretations of length and width of the ad. For example if it was a banner add on either sides of a page the width is more likely to be smaller than then length, hence value is less than one. When the ad is in the bottom or the top of a page, its width is lager than the length, hence the larger than 1 value. We'll leave this variable for now and observe its graph later.

```
In [17]: (ad_removed['ad_area']>1).value_counts()/187561
```

```
Out[17]: False    0.564739
         True     0.435261
         Name: ad_area, dtype: float64
```

We also removed the observation that have ctr larger than 2 during the process.

2.2.2 Categorical Features

All the categorical values appear to have the right set of values. However, the the deviceType, there are only 3 device options, there are 4 unique values.

```
In [18]: ad_removed['deviceType'].value_counts()

Out[18]: 2    76169
         1    73097
         3    36348
         5     1947
         Name: deviceType, dtype: int64
```

The value 5 is the smallest value; this could be a way to communicate unknown device. We can ignore these values since they contribute only about 1% of the remaining data.

```
In [19]: 2113/187561

Out[19]: 0.011265668235933909

In [20]: ad_removed['deviceType'] = ad_removed['deviceType'].astype('int64')

In [21]: ad_removed = ad_removed[ad_removed.deviceType <5]

In [22]: ad_removed['deviceType'] = ad_removed['deviceType'].astype('category')
```

In addtion, the variable viewability represent percetage of an ad is viewed therefore, all values should be smaller than 1

```
In [23]: ad_removed = ad_removed[ad_removed.viewability <=1]
```

We also will check the uniue value of these categorical values, just in case

```
In [24]: for col in categoricalColumn:
             print('Unique values for ' + col)
             print(ad_removed[col].unique())
             print('')

Unique values for companyId
[95, 43, 159, 40, 157, 126]
Categories (6, int64): [95, 43, 159, 40, 157, 126]

Unique values for countryId
[234, 57, 29, 70, 198, ..., 1, 92, 8, 217, 250]
Length: 163
Categories (163, int64): [234, 57, 29, 70, ..., 92, 8, 217, 250]

Unique values for deviceType
[1, 2, 3]
Categories (3, int64): [1, 2, 3]
```

```
Unique values for dow
[Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday]
Categories (7, object): [Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday]
```

3 Data Exploration

3.1 Univariate Visualisation

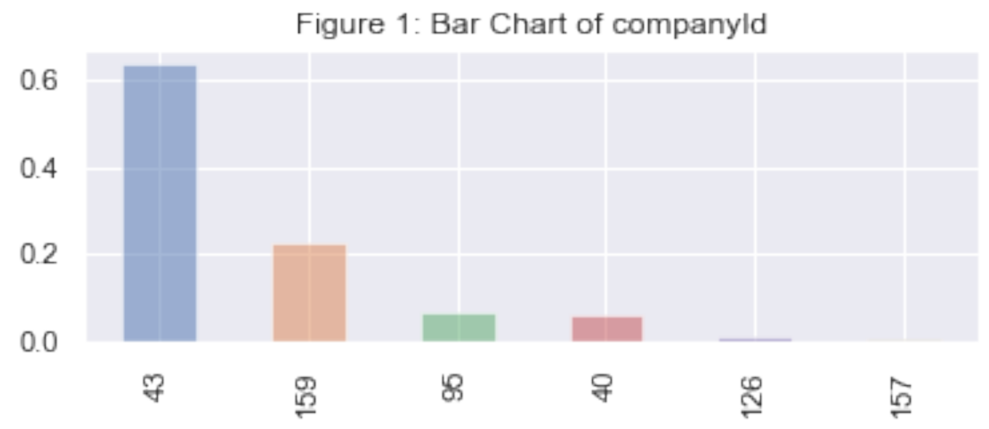
For convenience, we defined two functions named `BarPlot(x)` and `BoxHistogramPlot(x)` for categorical and numerical features respectively. For given an input categorical column `x`, `BarPlot(x)` returns a bar chart with percentage on top of each bar. A bar chart is useful to present the proportions by categories. For given an input numerical column `x`, `BoxHistogramPlot(x)` plots a histogram and a box plot. A histogram is useful to visualize the shape of the underlying distribution whereas A box plot tells the range of the attribute and helps detect any outliers. The following chunk codes show how these function were defined using the numpy library and the matplotlib library.

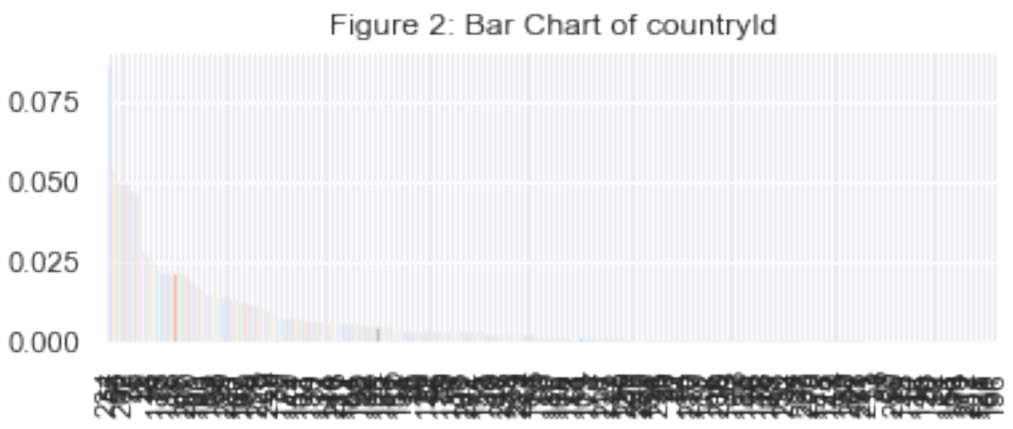
```
In [25]: import matplotlib.pyplot as plt
         import seaborn as sns

         sns.set(color_codes=True)

         def BarPlot(x):
             total = float(len(ad_removed))
             ax = ad_removed[x].value_counts(normalize = True).plot(
                 kind = "bar", alpha = 0.5)

         def BoxHistogramPlot(x):
             f, (ax_box, ax_hist) = plt.subplots(2, sharex=True,
                                                 gridspec_kw={"height_ratios": (.15, .85)})
             sns.boxplot(x, ax=ax_box)
             sns.distplot(x, ax=ax_hist)

             ax_box.set(yticks=[])
             sns.despine(ax=ax_hist)
             sns.despine(ax=ax_box, left=True)
             plt.show()
```

For companyId variable, majority of the data is from comnyId=43. It is hard to identify which countries are the most popular based on the graph. The device 1 and 2 are used the most, they are potentially desktop and mobile respectively. The data spread out quite evenly throughout the week, with the weekend is slightly more .

```
In [26]: i = 1    # initialize figure labelling4)
         for col in ['companyId', 'countryId', 'deviceType', 'dow']:
```

```
plt.figure(figsize=(6,2))
plt.title("Figure " + str(i) + ": Bar Chart of " + col, fontsize = 12)
BarPlot(col)
plt.show()
i = i + 1
```





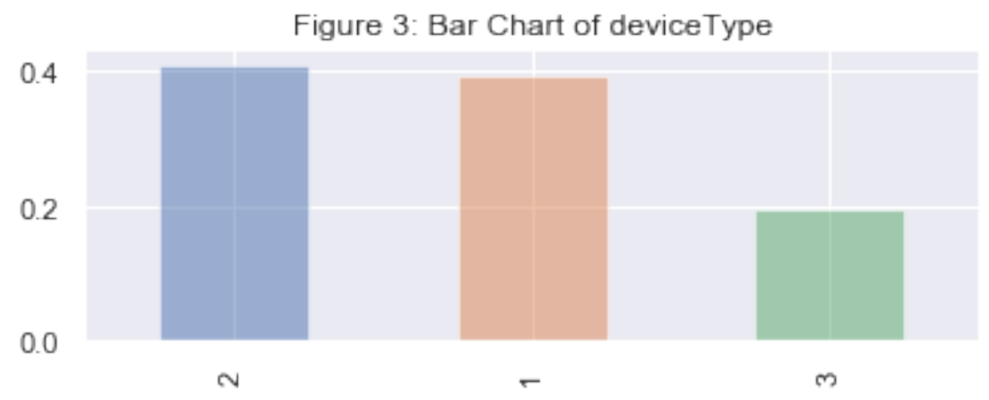
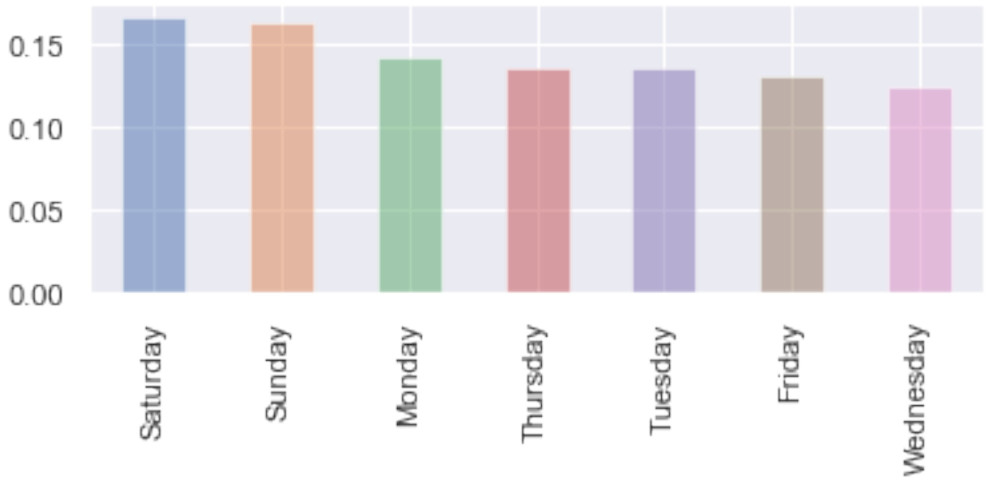
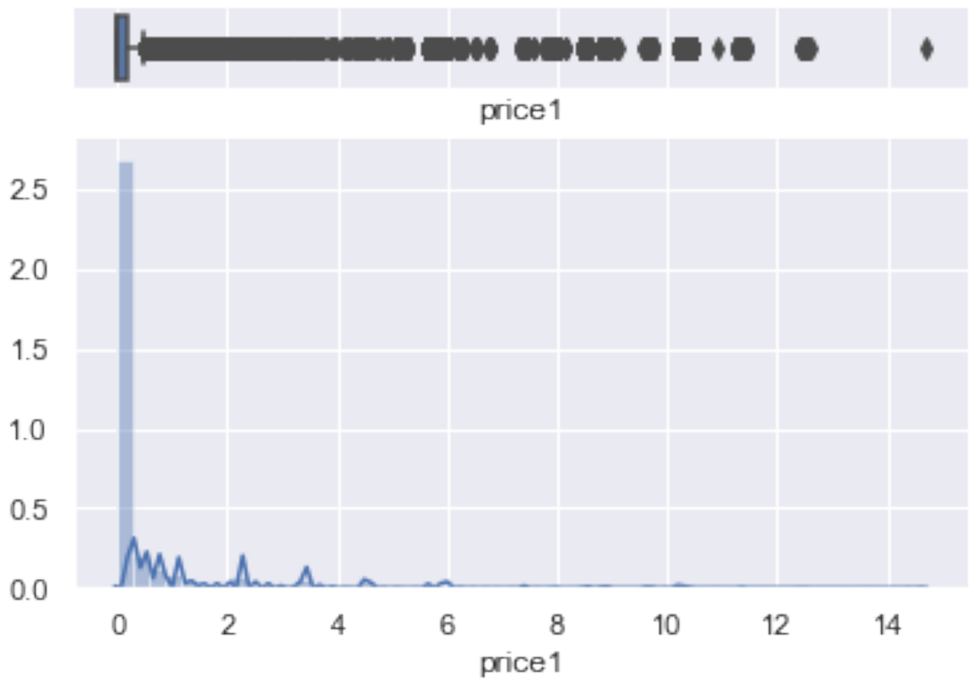


Figure 4: Bar Chart of dow

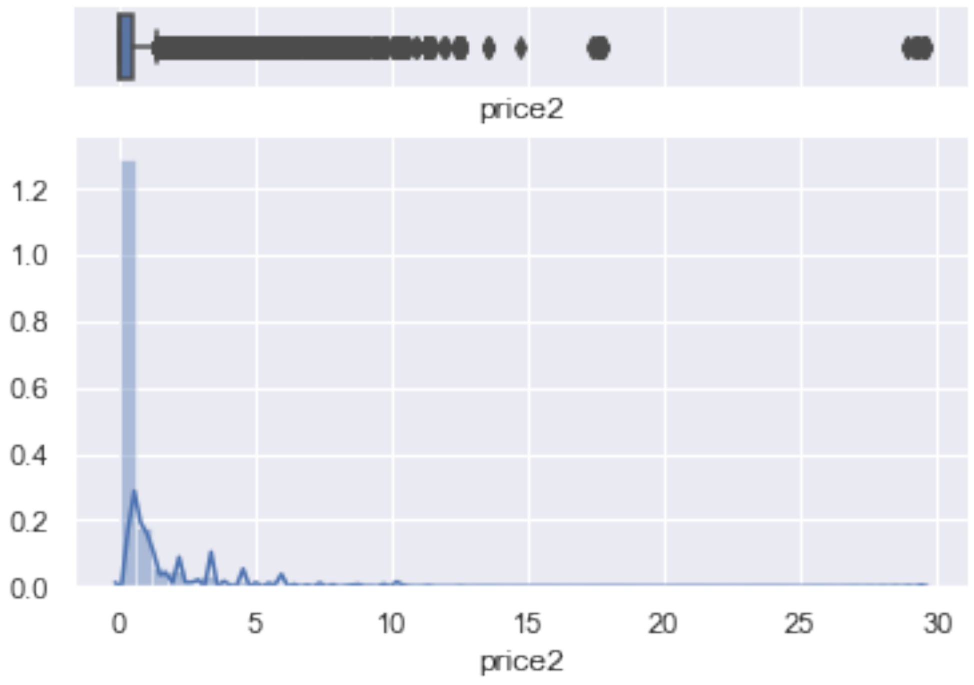


```
In [27]: for col in ['price1', 'price2','price3','ad_area', 'ad_ratio','requests','impression'
             plt.figure(figsize=(6,2))

             plt.suptitle("Figure " + str(i) + ": Histogram and Box Plot of " + col)
             BoxHistogramPlot(ad_removed[col])
             plt.show()
             i = 1 + i
```

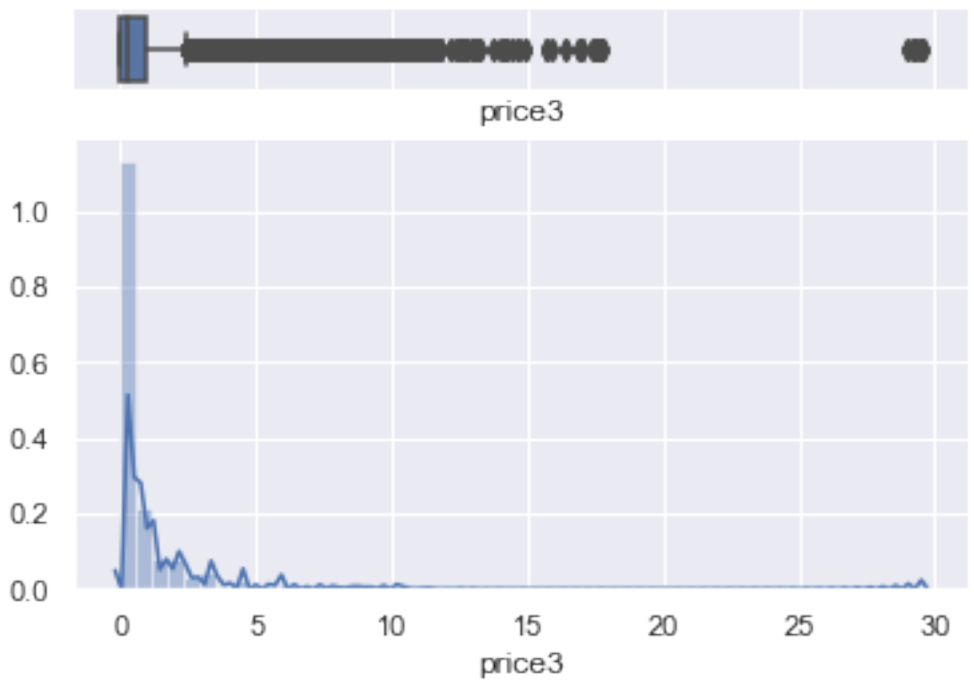
```
<Figure size 432x144 with 0 Axes>
```


price1
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
14
price1

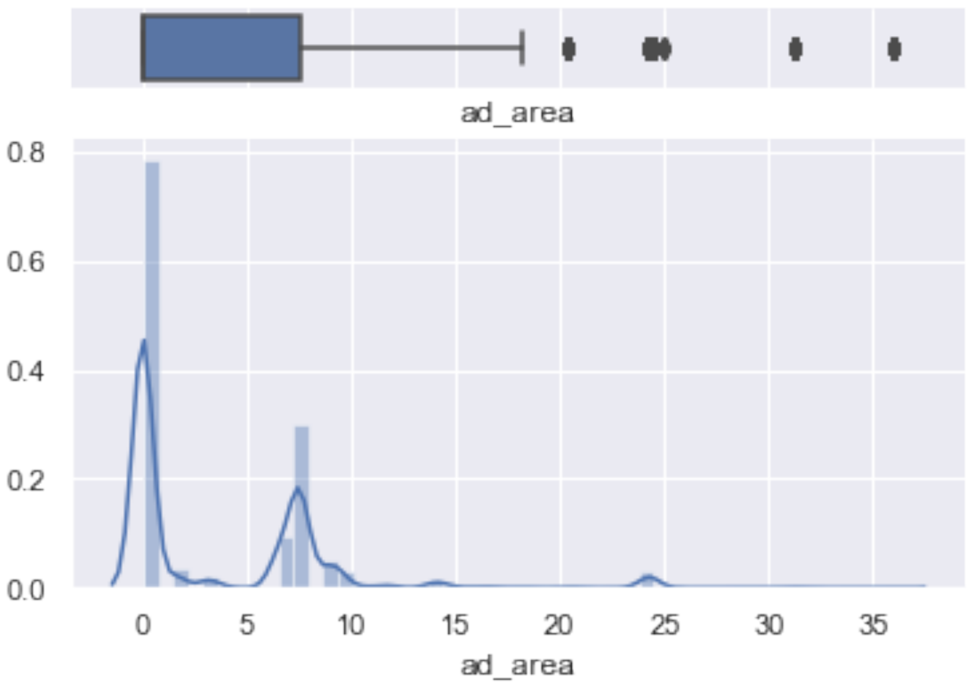
```
<Figure size 432x144 with 0 Axes>
```


price2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
price2

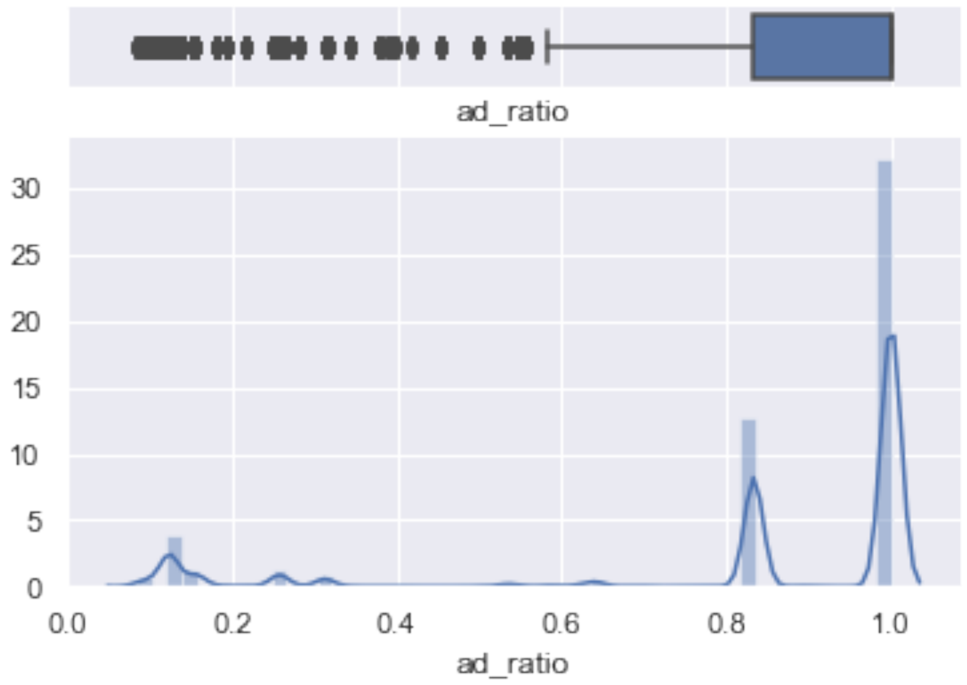
```
<Figure size 432x144 with 0 Axes>
```


price3
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
price3

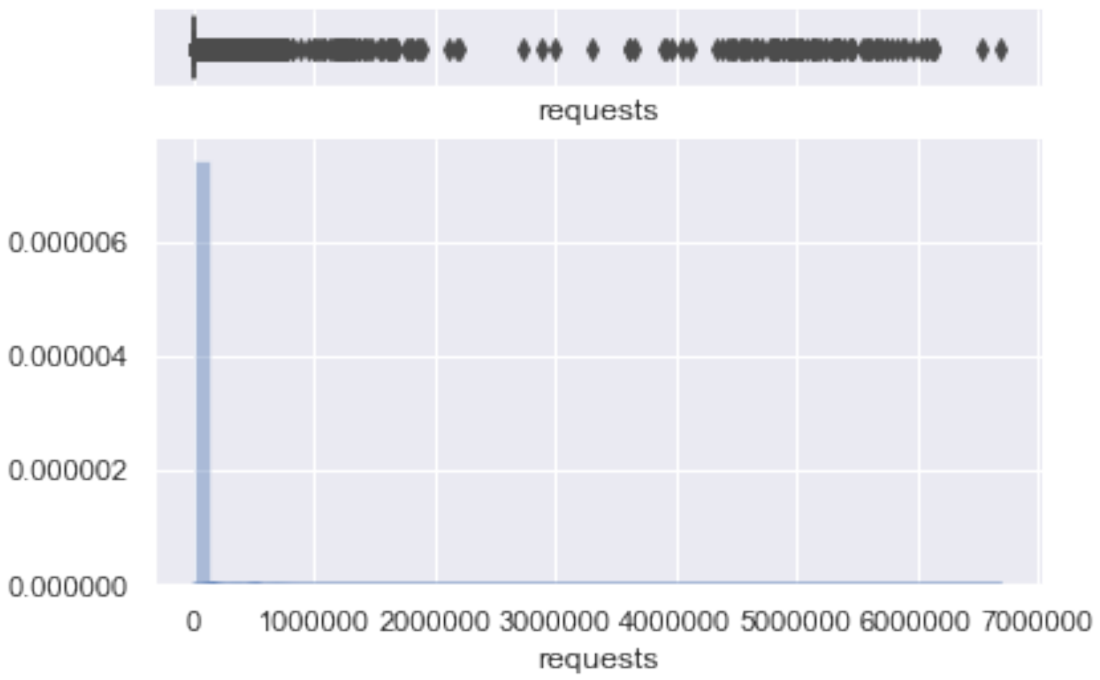
```
<Figure size 432x144 with 0 Axes>
```


ad_area
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
ad_area

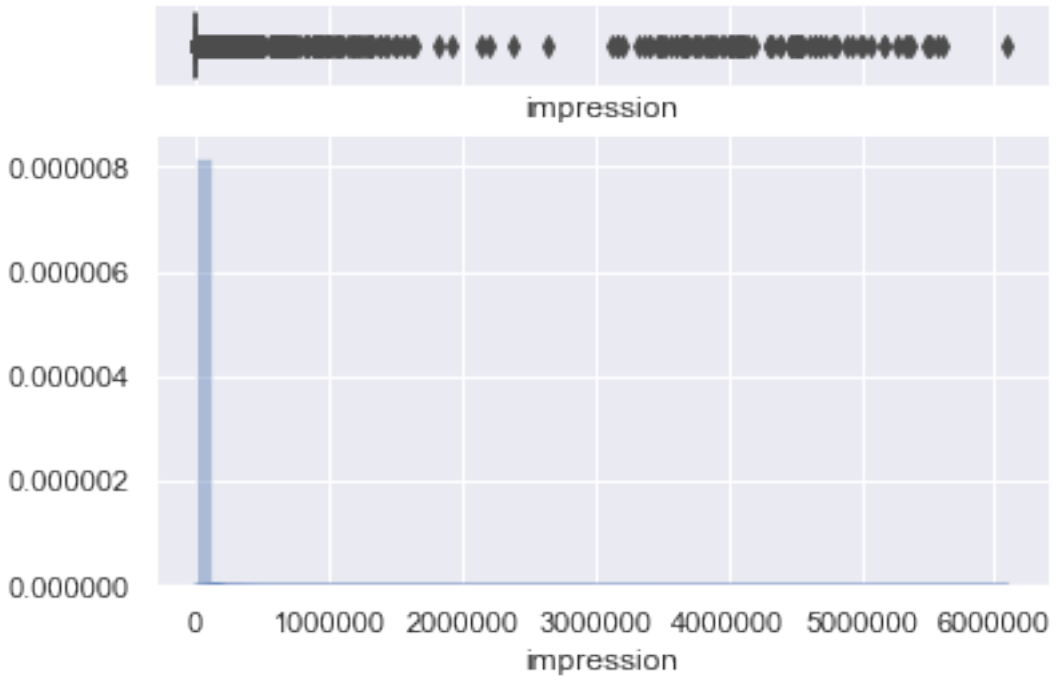
```
<Figure size 432x144 with 0 Axes>
```


ad_ratio
30
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
ad_ratio

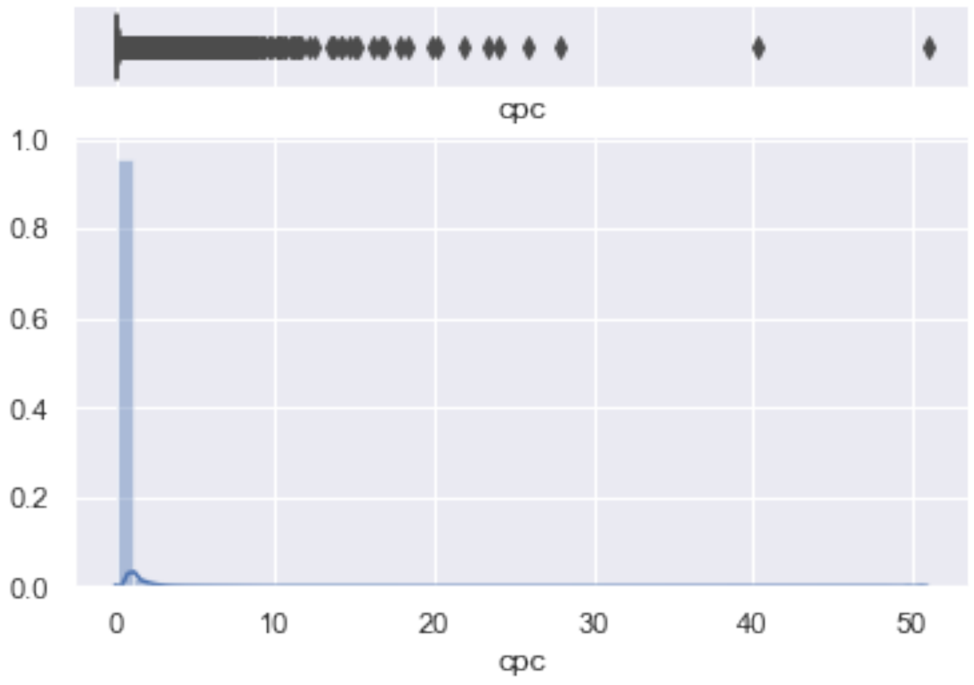
```
<Figure size 432x144 with 0 Axes>
```


requests
0.000006
0.000004
0.000002
0.000000
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000
requests

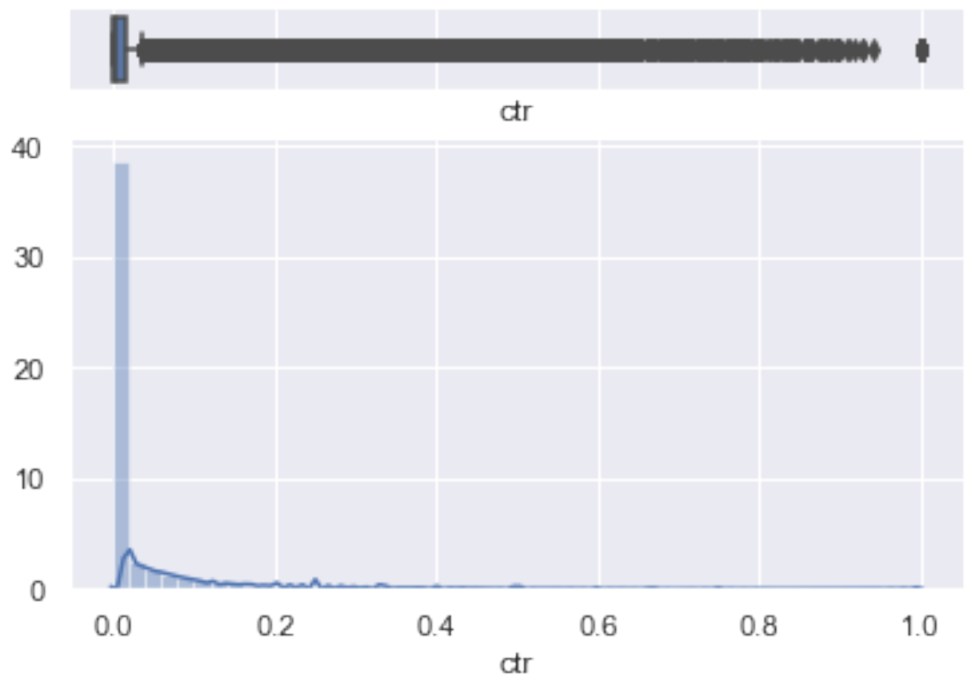
```
<Figure size 432x144 with 0 Axes>
```


impression
0.000008
0.000006
0.000004
0.000002
0.000000
0
1000000
2000000
3000000
4000000
5000000
6000000
impression

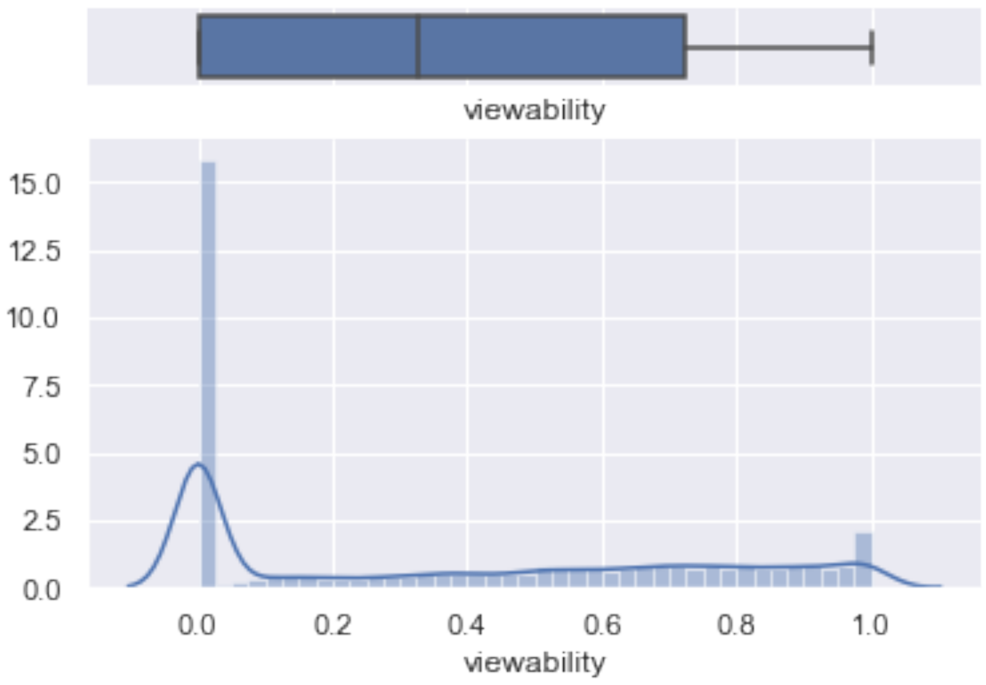
```
<Figure size 432x144 with 0 Axes>
```


cpc
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
cpc

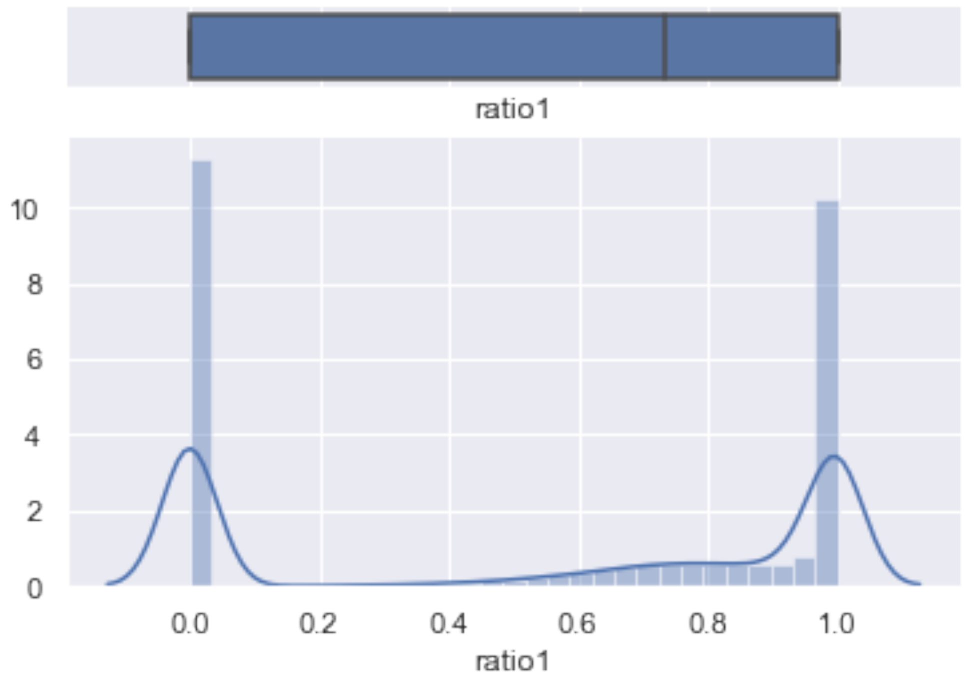
```
<Figure size 432x144 with 0 Axes>
```


ctr
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
ctr

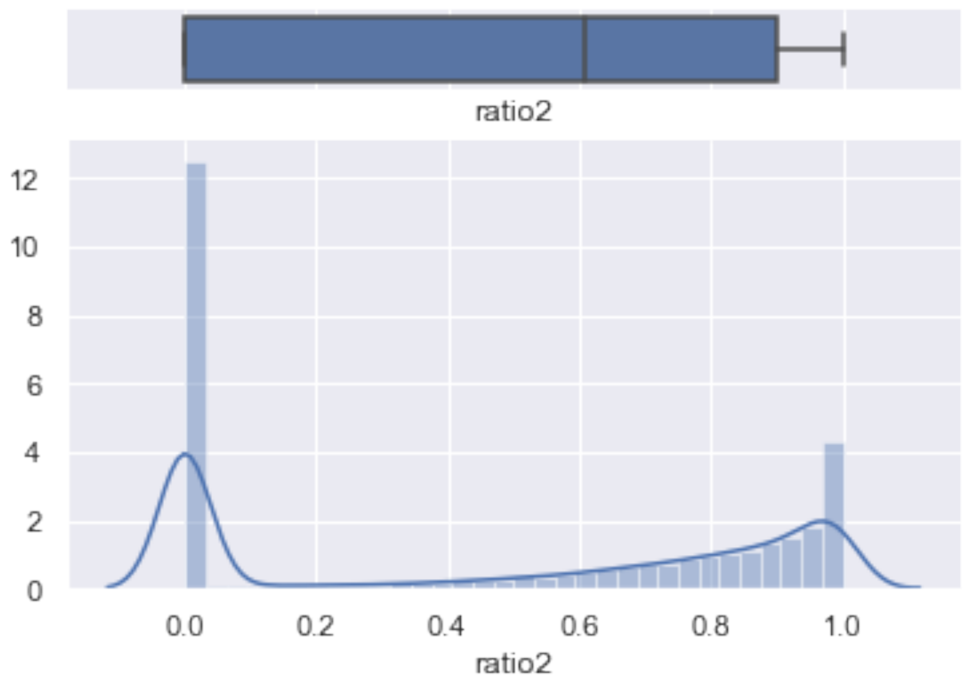
```
<Figure size 432x144 with 0 Axes>
```


viewability
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
viewability

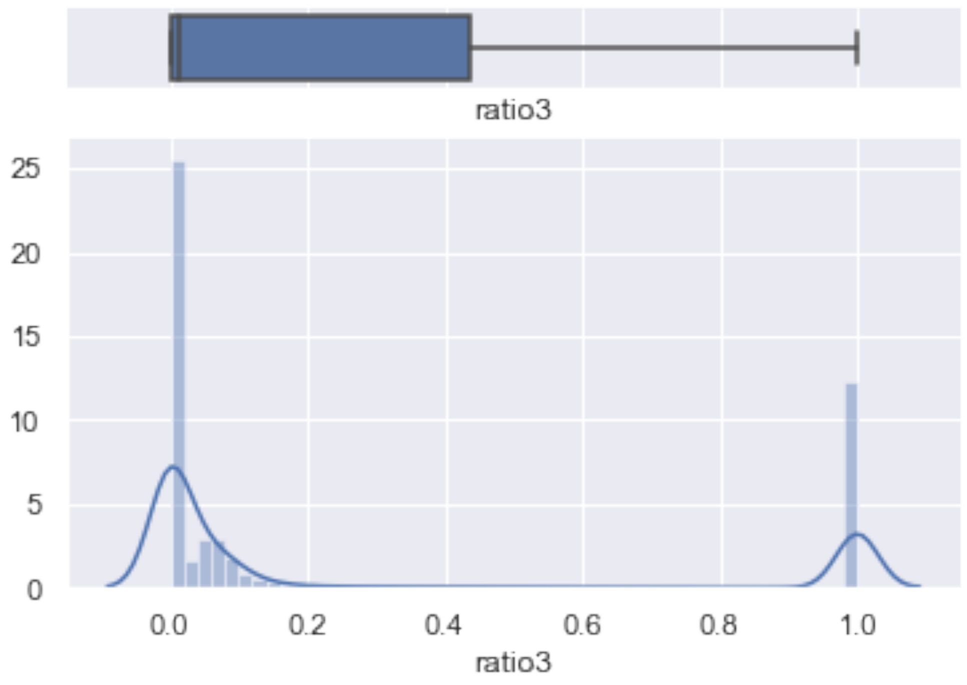
```
<Figure size 432x144 with 0 Axes>
```


ratio1
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0
ratio1

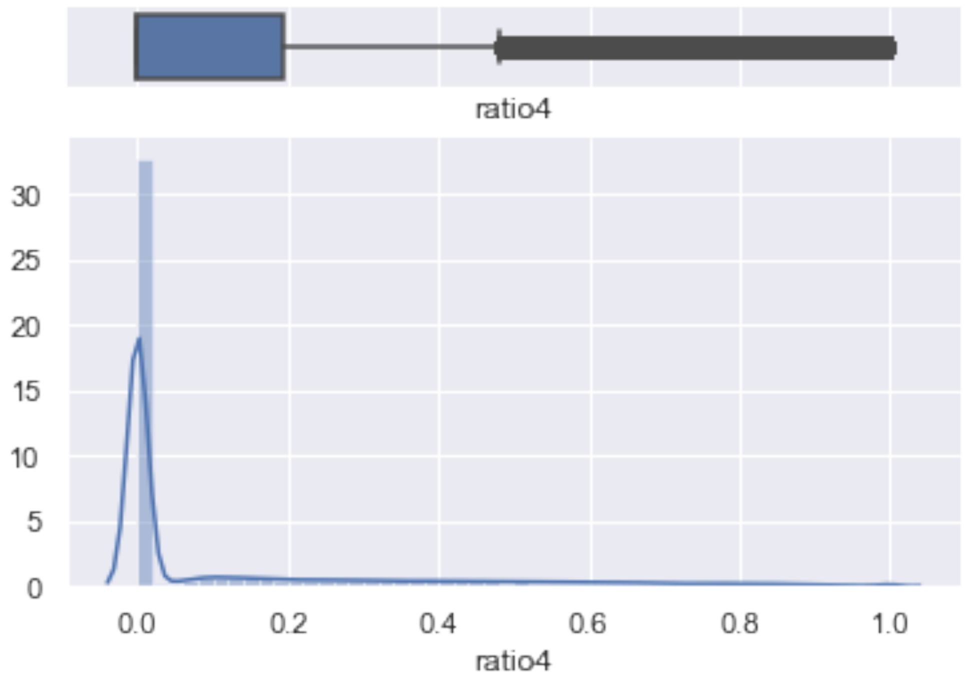
```
<Figure size 432x144 with 0 Axes>
```


ratio2
12
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0
ratio2

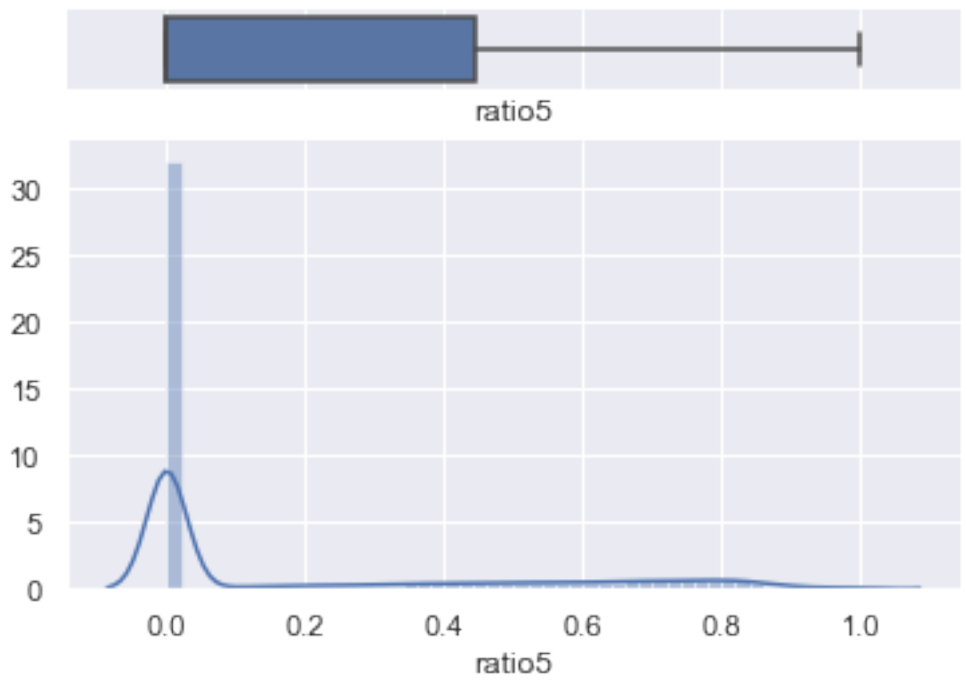
```
<Figure size 432x144 with 0 Axes>
```


ratio3
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
ratio3

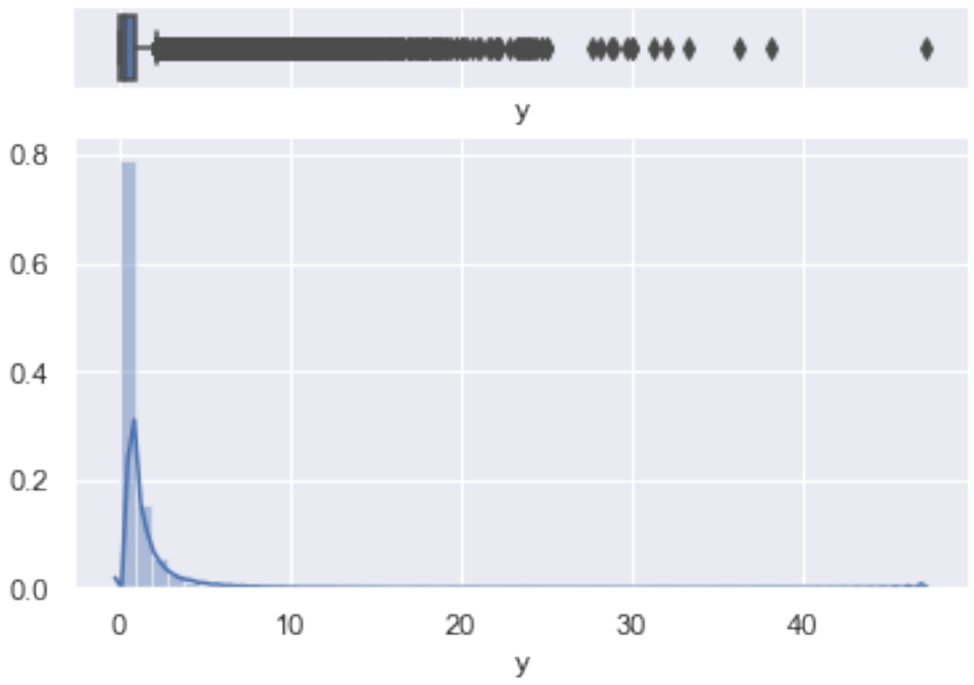
```
<Figure size 432x144 with 0 Axes>
```


ratio4
30
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
ratio4

```
<Figure size 432x144 with 0 Axes>
```


ratio5
30
25
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0
ratio5

```
<Figure size 432x144 with 0 Axes>
```



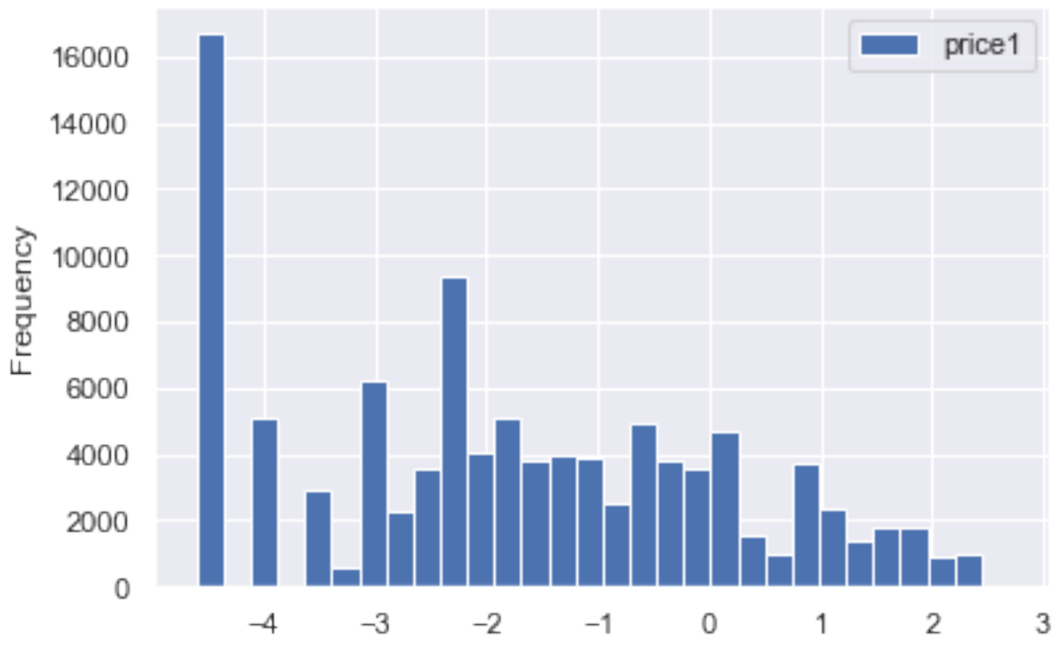
We noticed that the price1, price2 and price3 variables have a lot of 0 value. This would not be beneficial for modelling process if all these 3 variables are equal to zero at the same time. Because, in reality ads has to cost something therefore, we should remove observations with all these 3 variables equal to zero. The value of zero may indicate either organic reach or missing price value. Since some algorithm can deal with NA values we replace 0 price with NA, further processing will be considered for models that cannot handle missing values. We do this in order to keep as many observations as possible.

```
In [28]: ad_removed.loc[ad_removed.price1==0,'price1']=np.nan

In [29]: ad_removed.loc[ad_removed.price2==0,'price2']=np.nan

In [30]: ad_removed.loc[ad_removed.price3==0,'price3']=np.nan

In [31]: ad_removed['price1'].describe()

Out[31]: count    102455.000000
         mean          0.810484
         std           1.659465
         min           0.010000
         25%           0.050000
         50%           0.150000
         75%           0.740000
         max          14.690000
         Name: price1, dtype: float64
```

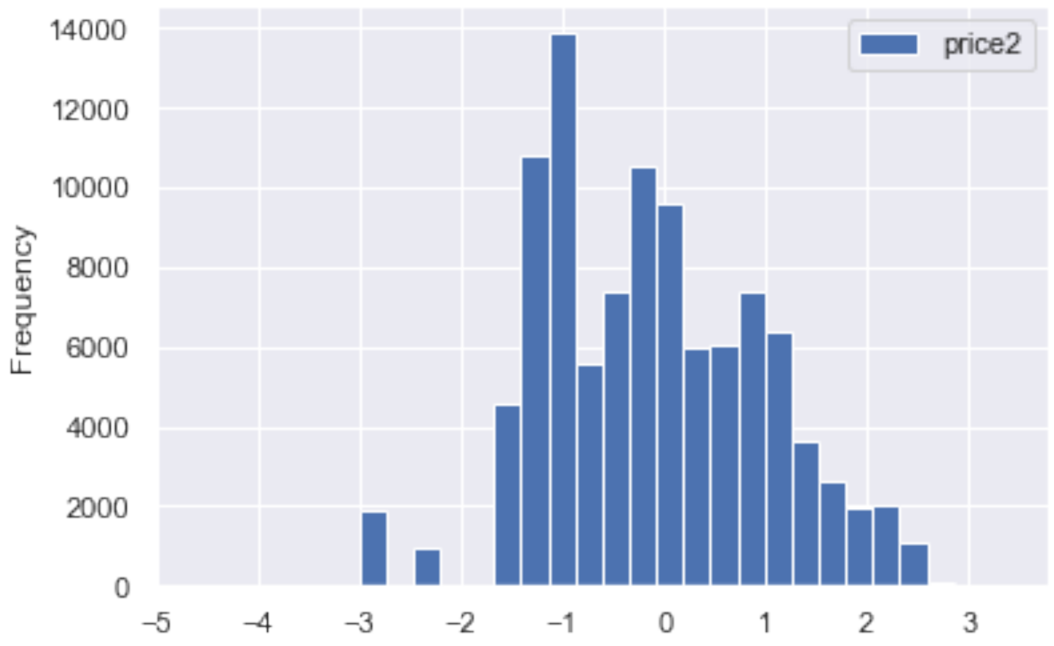
We can apply log transformation to these variables since their distritutions are right-skewed the we can observed their distribution again.

```
In [32]: ad_removed["price1"]=ad_removed["price1"].apply(np.log)
         ad_removed["price2"]=ad_removed["price3"].apply(np.log)
         ad_removed["price3"]=ad_removed["price3"].apply(np.log)
```

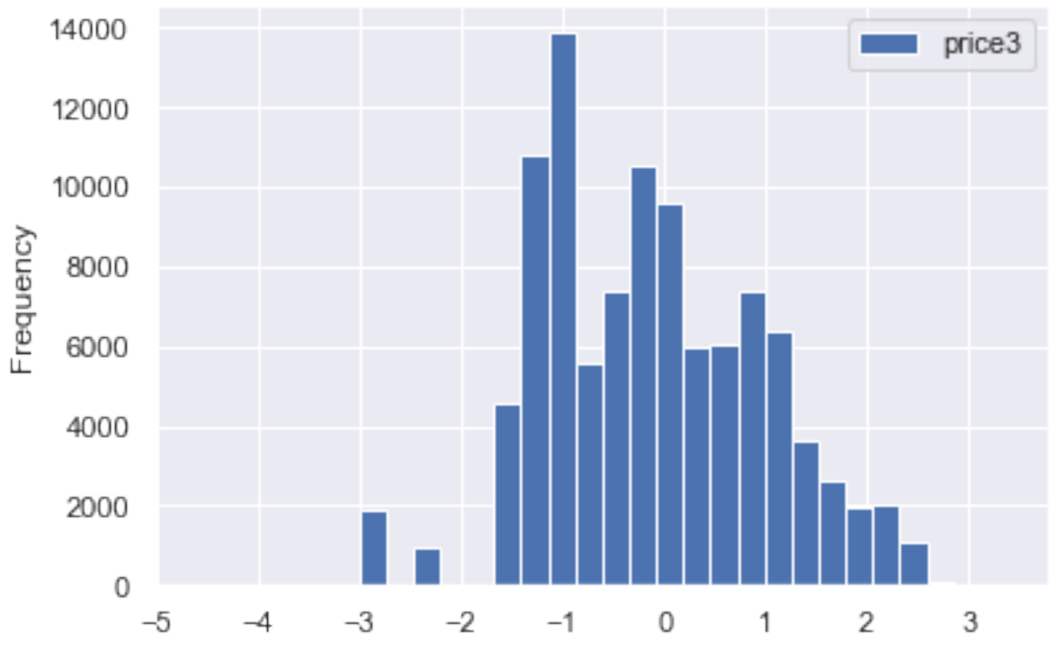
```
In [33]: ad_removed['price1'].plot(kind='hist',bins=30)
         plt.legend()
         plt.show()
```



```
In [34]: ad_removed['price2'].plot(kind='hist',bins=30)
         plt.legend()
         plt.show()
```



```
In [35]: ad_removed['price3'].plot(kind='hist',bins=30)
         plt.legend()
         plt.show()
```



We can see that the distributions of these three variables are much closer to a normal distribution after the log transformation, ignoring the zero values.

```
In [36]: from scipy.stats import boxcox
```

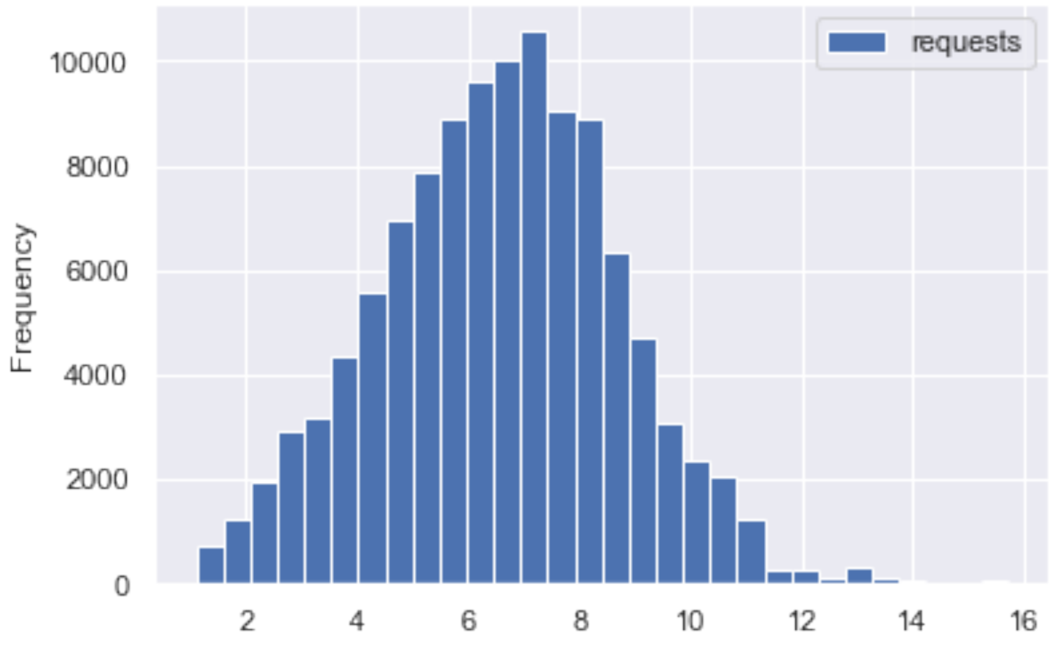
According to Google Ads, "An ad request is counted whenever your site requests ads to be displayed. It is the number of ad units that requested ads. We report an ad request each time a request was sent, even if no ads were returned and backfill ads were displayed instead." and "An impression is counted each time an ad loads on a site. If you refresh the page, even if the same ad loads, a new impression is counted." Therefore, it is reasonable to exclude the zero value for the modelling process; since value zero here does not have any predicting value for advertisers. We use the same approach as the prices values.

```
In [37]: ad_removed.loc[ad_removed.requests==0,'requests']=np.nan
```

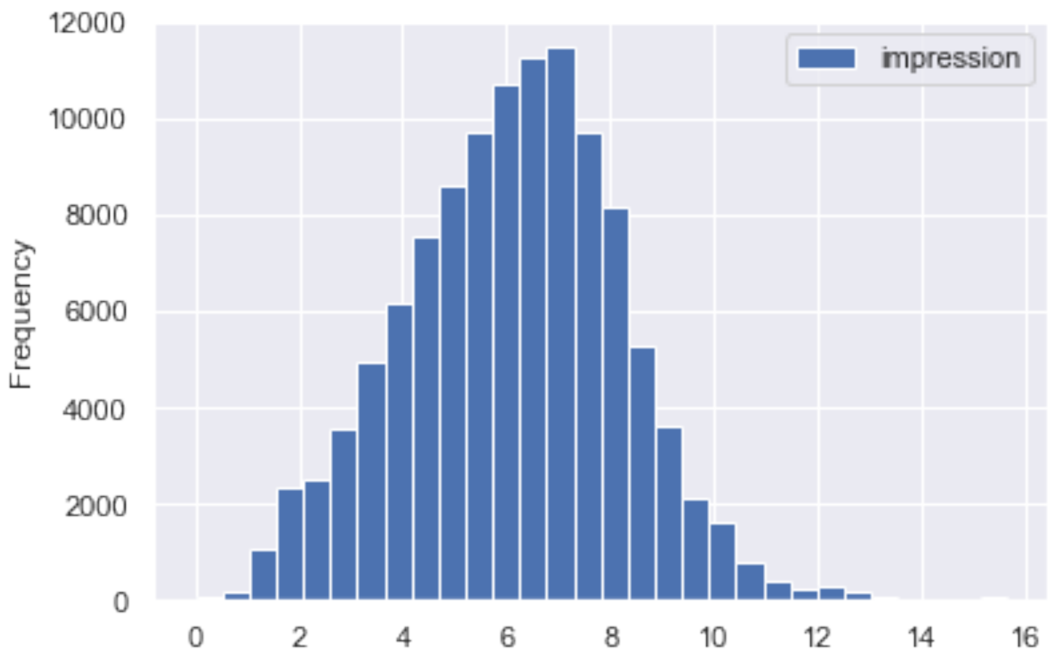
```
In [38]: ad_removed.loc[ad_removed.impression==0,'impression']=np.nan
```

```
In [39]: from scipy.stats import boxcox
```

```
In [40]: ad_removed.requests= boxcox(ad_removed.requests, 0)
         ad_removed['requests'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```



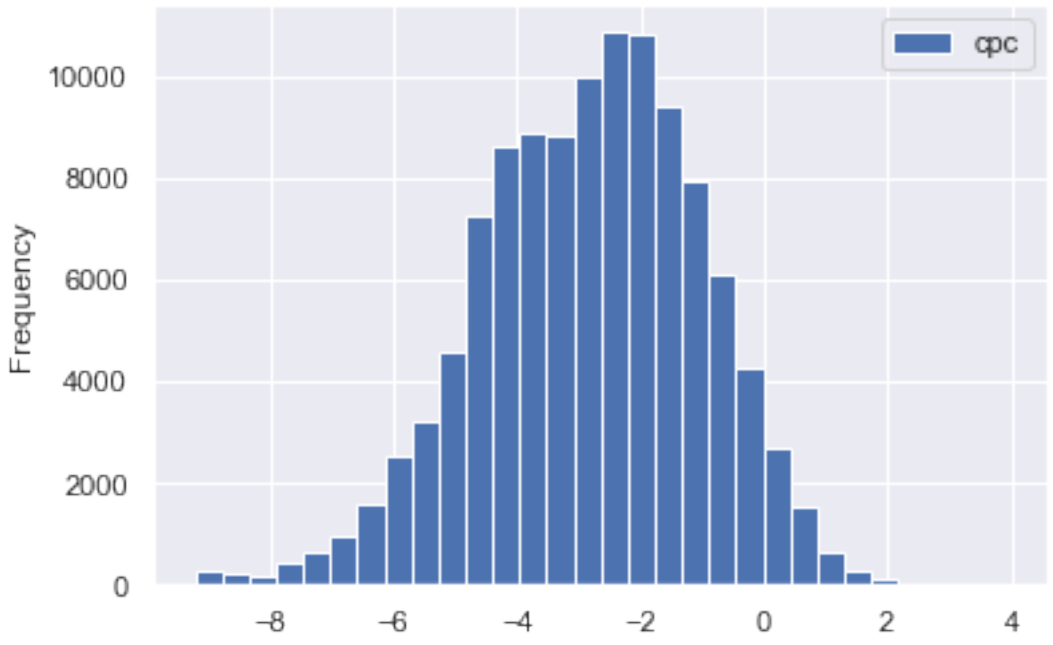
```
In [41]: ad_removed.impression= boxcox(ad_removed.impression, 0)
         ad_removed['impression'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```



For cpc: cost per click values, the same logic can be applied.

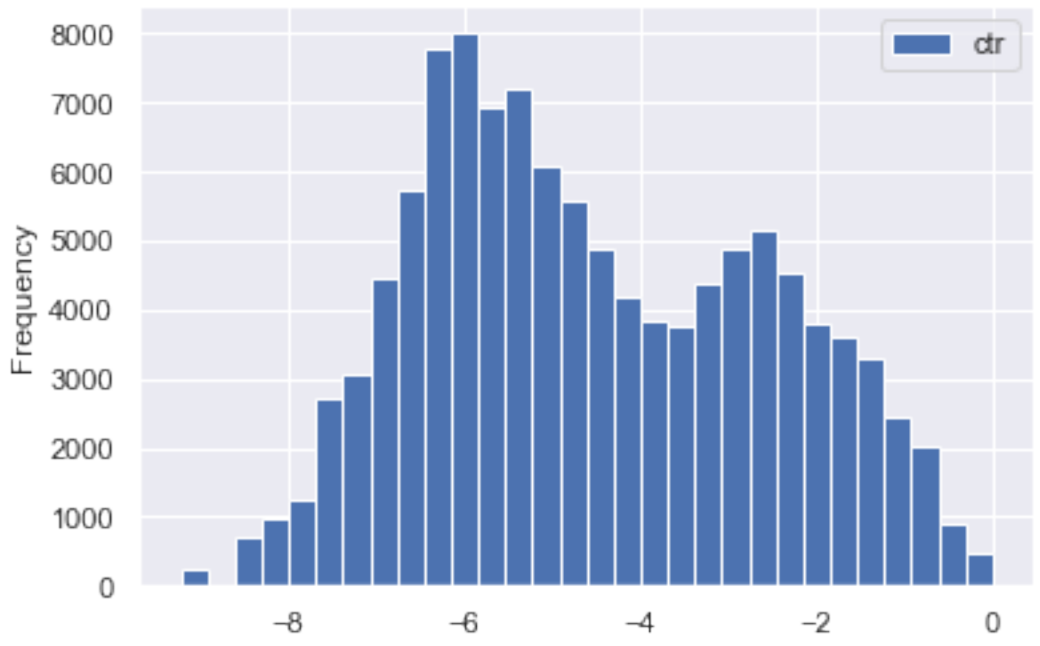
```
In [42]: ad_removed.loc[ad_removed.cpc==0,'cpc']=np.nan
```

```
In [43]: ad_removed.cpc= boxcox(ad_removed.cpc, 0)
         ad_removed['cpc'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```

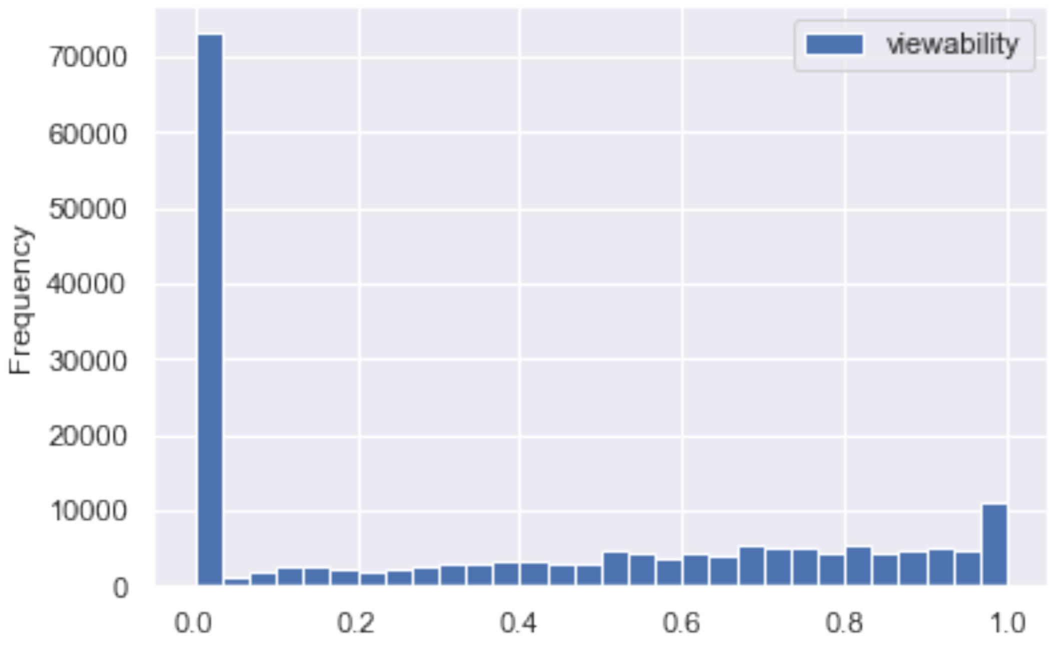


For ctr: click through rate and viewability value of zero contain predictive values. For example, some ads can have millions of impressions but no one clicks on it, this will male ctr equals to zero. Similarly, when a user scrolls through a page the visibility of an add can be zero. In oreder, to use transformation for these variables, add a constant, 1 to all the values.

```
In [44]: ad_removed.loc[ad_removed.ctr==0,'ctr']=np.nan
         #ad_removed["ctr"]=ad_removed["ctr"].apply(np.log)
         ad_removed.ctr= boxcox(ad_removed.ctr, 0)
         ad_removed['ctr'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```



```
In [45]: ad_removed['viewability'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```



The viewability is dominated by 0, therefore, we chose to create a new categorical vari-ble"adSeen" based on the viewability. 0 is "not seen", (0,0.5] is "partly seen" and (0.5,1] is "seen"

```
In [46]: ad_removed['adSeen'] = pd.cut(ad_removed['viewability'], bins=[float('-Inf'),0, 0.5,
```

```
In [ ]:
```

We now need to address the variable ad_area. Almost half of the data have larger than 1 values; this could be the products of wrongly label the ratio between width and length. For example, a banner ad would have width that larger than length and a half-page ad will have width to length ratio less than 1. If this assumption is correct we can use the Google Ads size guideline from https://support.google.com/adsense/answer/6002621?hl=en to copy up with new new variable "addType". The range is based on the histogram of the variable ad_area and the top performing ad sizes according to Google.

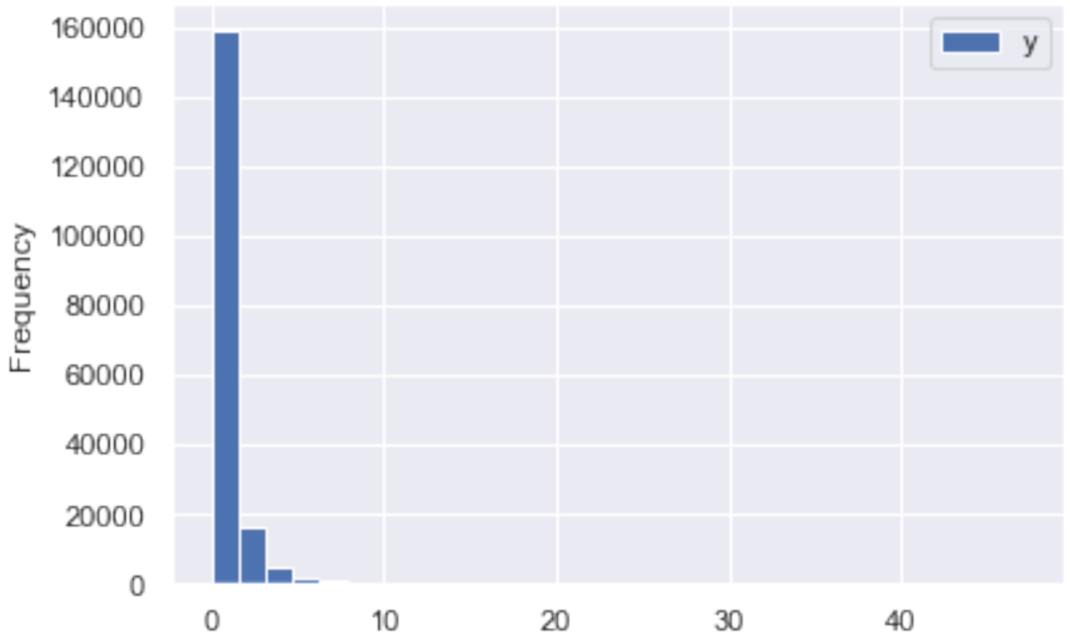
```
In [47]: ad_removed['adType'] = pd.cut(ad_removed['ad_area'], bins=[float('-Inf'),1.2,6,float(
```

```
In [48]: ad_removed['adType'].value_counts()
```

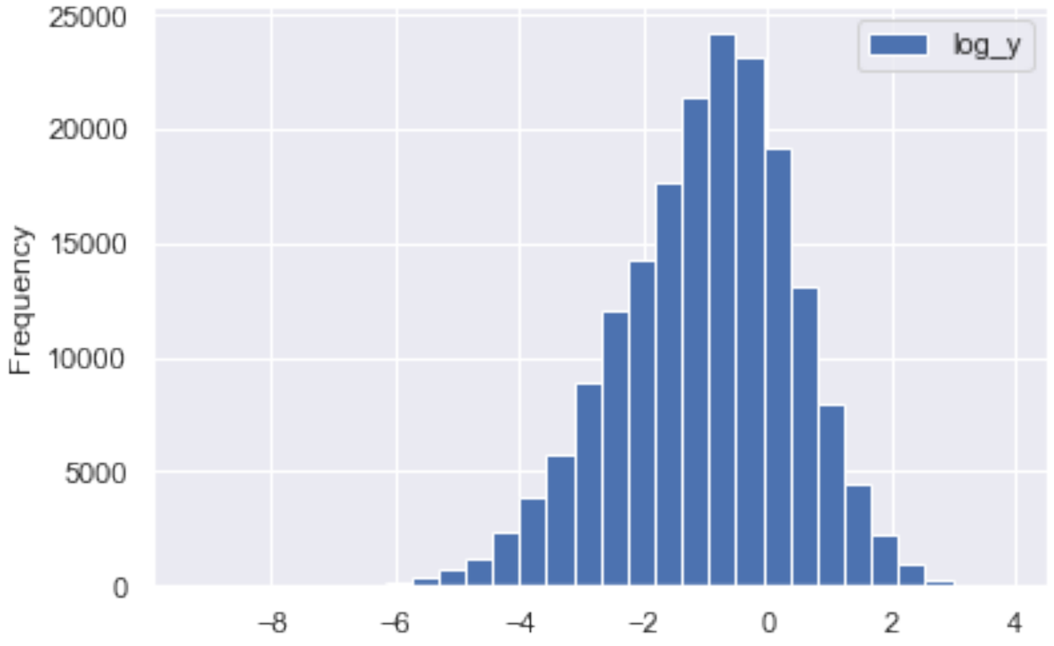
```
Out[48]: half-page      104048
         banner          72737
         regtangular      7893
         Name: adType, dtype: int64
```

Since there is not much information regarding the ratio variables, as long as they are within the stated normalised range. We'll leave them for now; this is a result of the lack of the domain knowledge regrading the data.

```
In [49]: ad_removed['y'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```



```
In [50]: ad_removed["log_y"]=ad_removed["y"].apply(np.log)

In [51]: ad_removed['log_y'].plot(kind='hist',bins=30)

         plt.legend()
         plt.show()
```

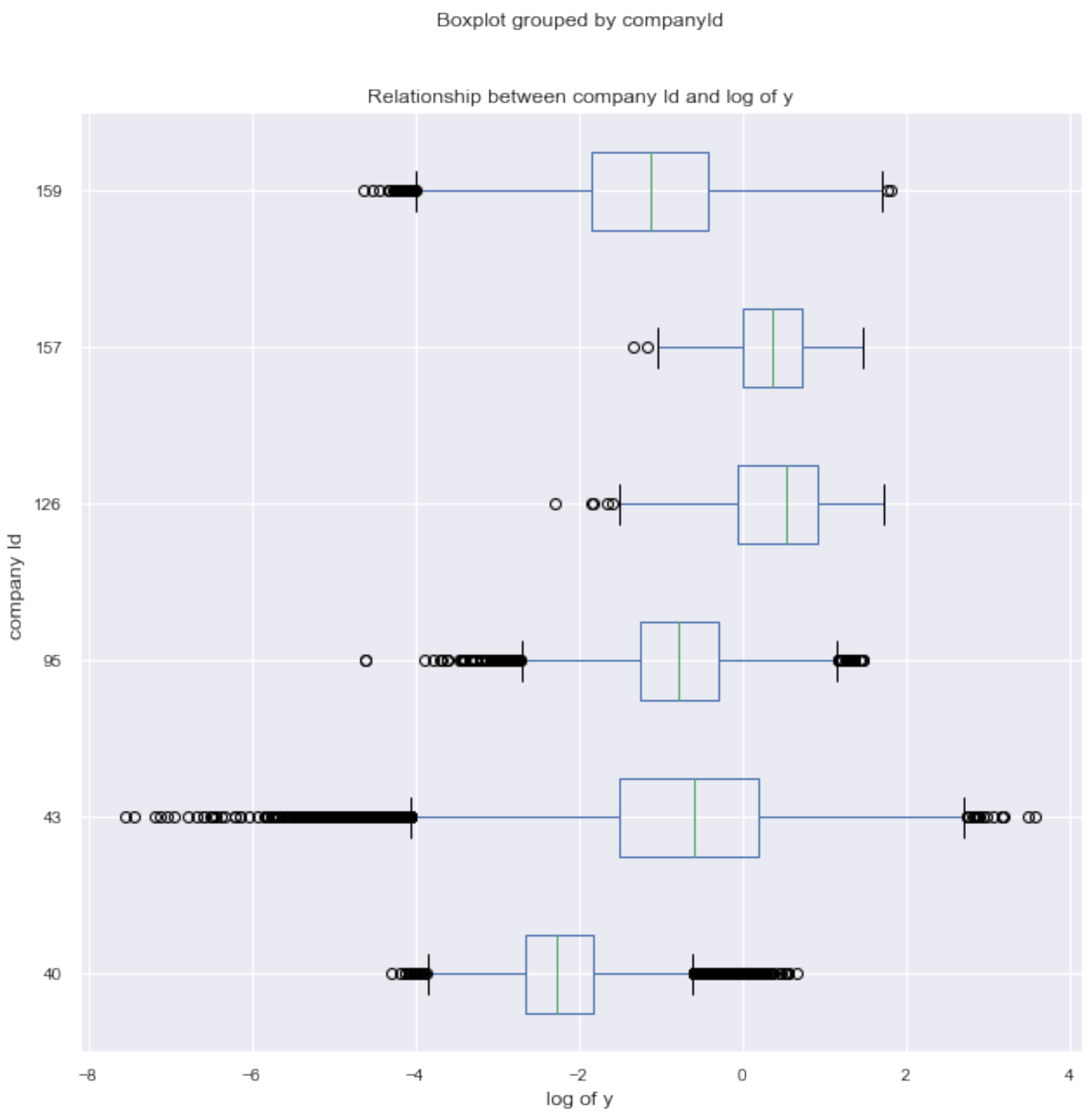


The target variable is normally distributed after a log transformation.

3.2 Multivariate Visualisation

3.2.1 Interaction between Categorical features and Target Feature

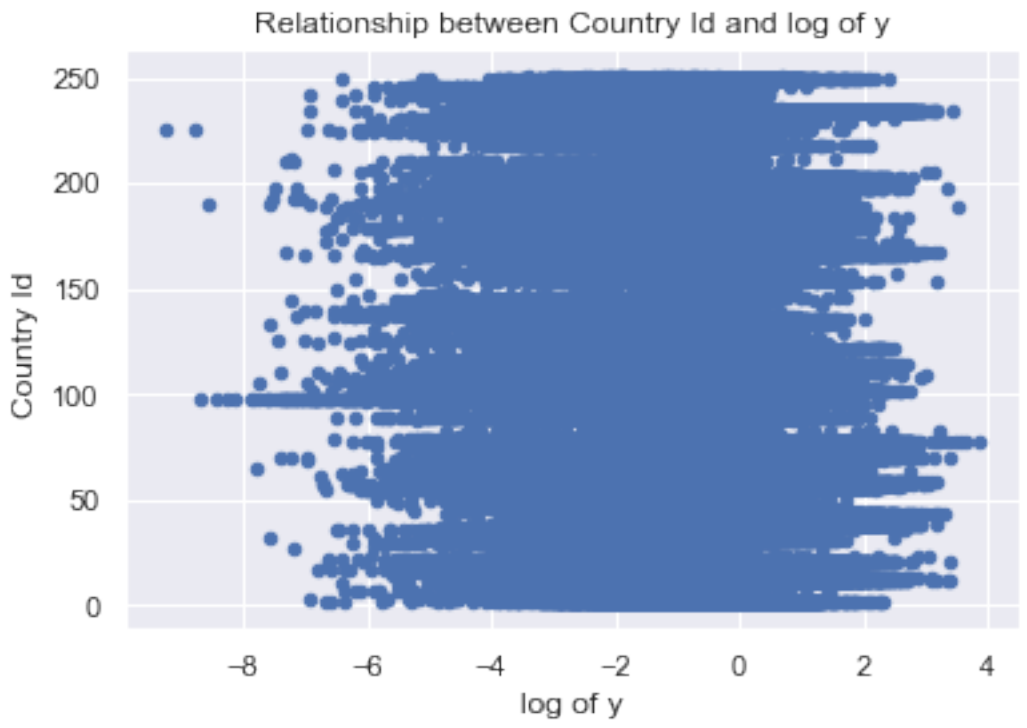
```
In [52]: ad_removed.dropna().boxplot(column='log_y',by='companyId',vert=False, figsize=(10,10)
         plt.xlabel('log of y')
         plt.ylabel('company Id ')
         plt.title('Relationship between company Id and log of y')
         plt.show()
```



We can see that companyId 157 and 126 achives more consistency when it comes to log of y; this infers that they may have more effective strategy in ad spent.

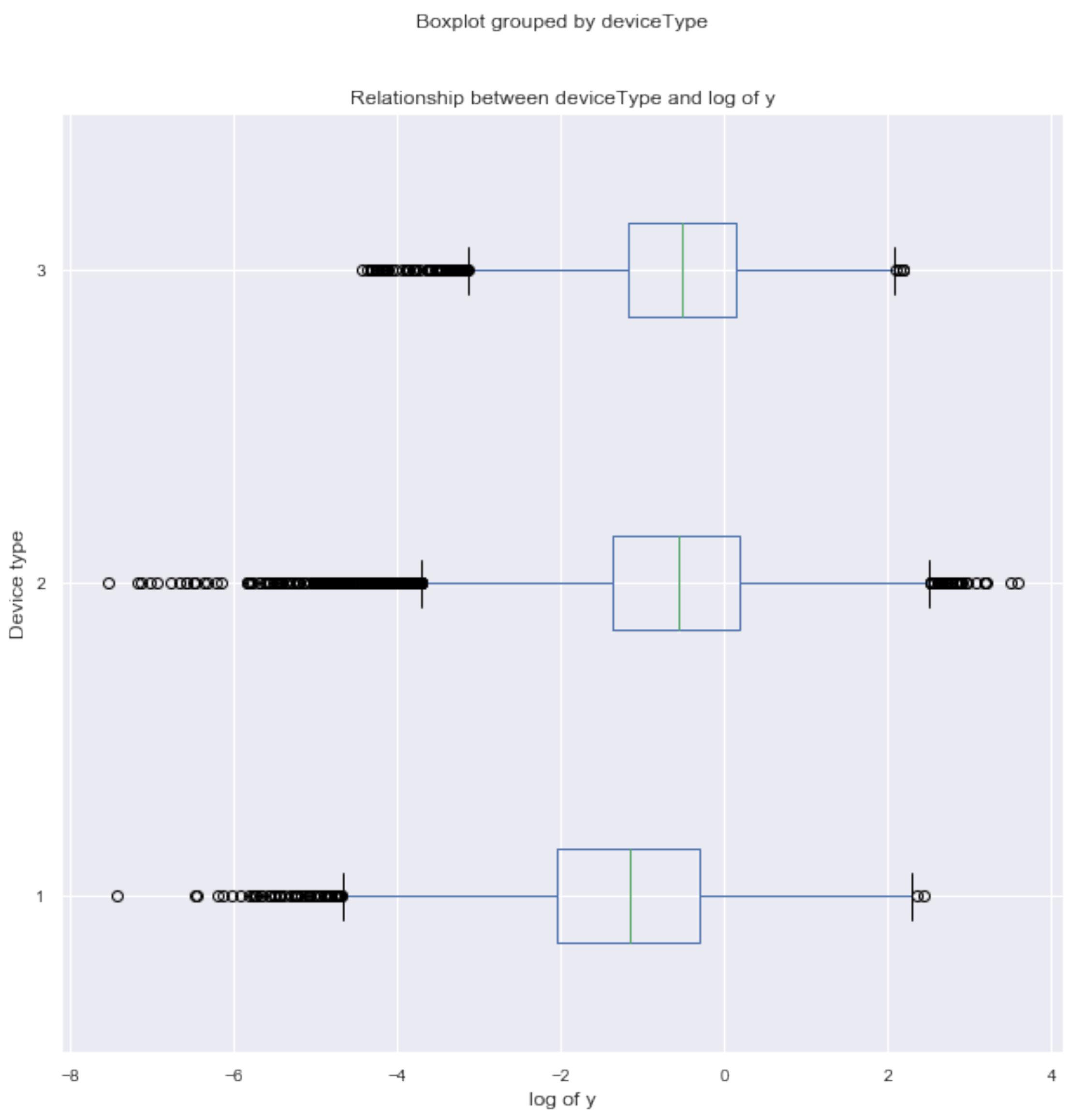
```
In [53]: ad_removed['countryId'] = ad_removed['countryId'].astype('int64')
         ad_removed.plot.scatter(y='countryId', x='log_y')
         plt.xlabel('log of y')
         plt.ylabel('Country Id ')
         plt.title('Relationship between Country Id and log of y')
         plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```



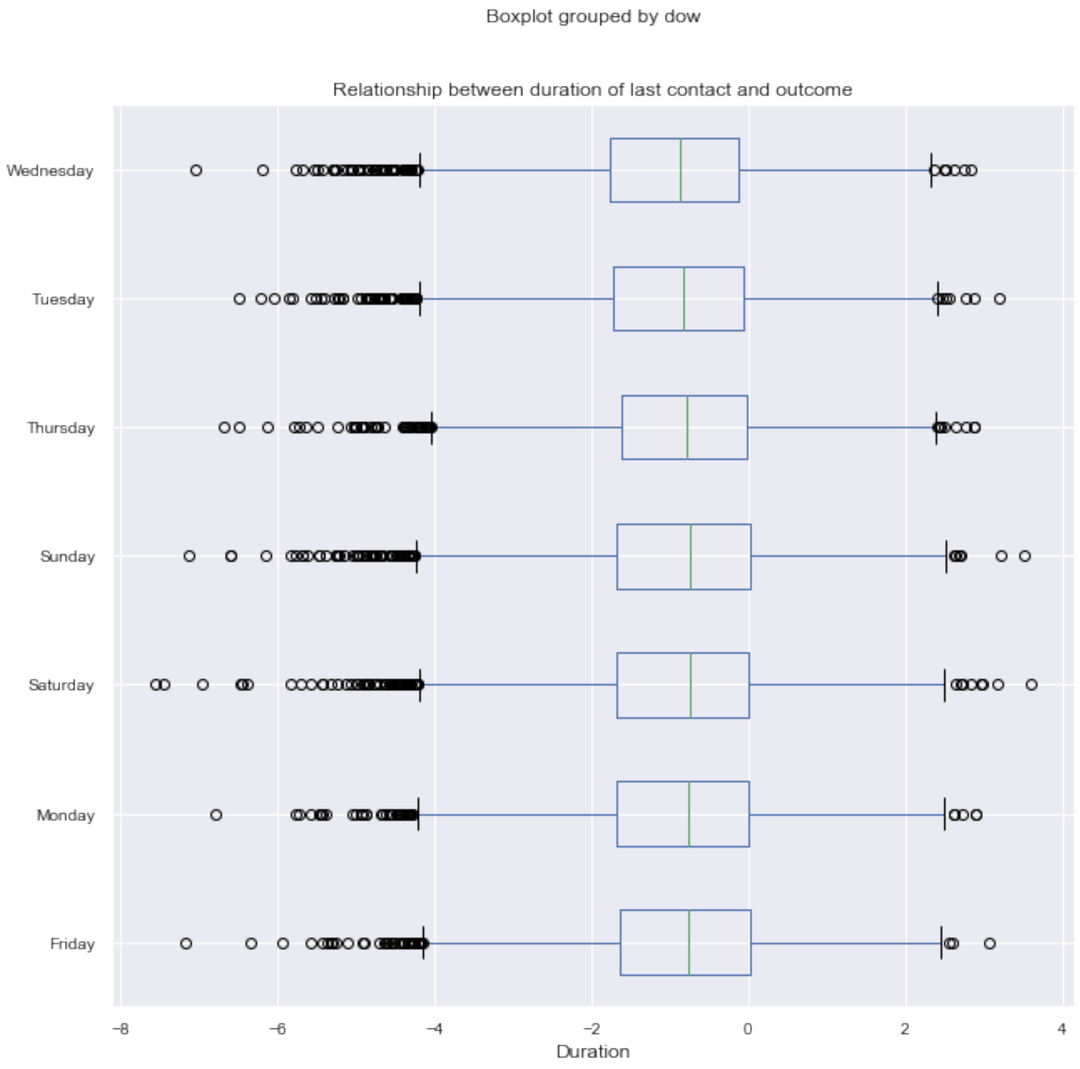
Once again, there is not much information regarding the countryId and the (log of) the target variable. Some countries seem to have more consistent (log of) y values than others, however, its not very clear.

```
In [54]: ad_removed.dropna().boxplot(column='log_y',by='deviceType',vert=False, figsize=(10,10
         plt.xlabel('log of y')
         plt.ylabel('Device type')
         plt.title('Relationship between deviceType and log of y')
         plt.show()
```



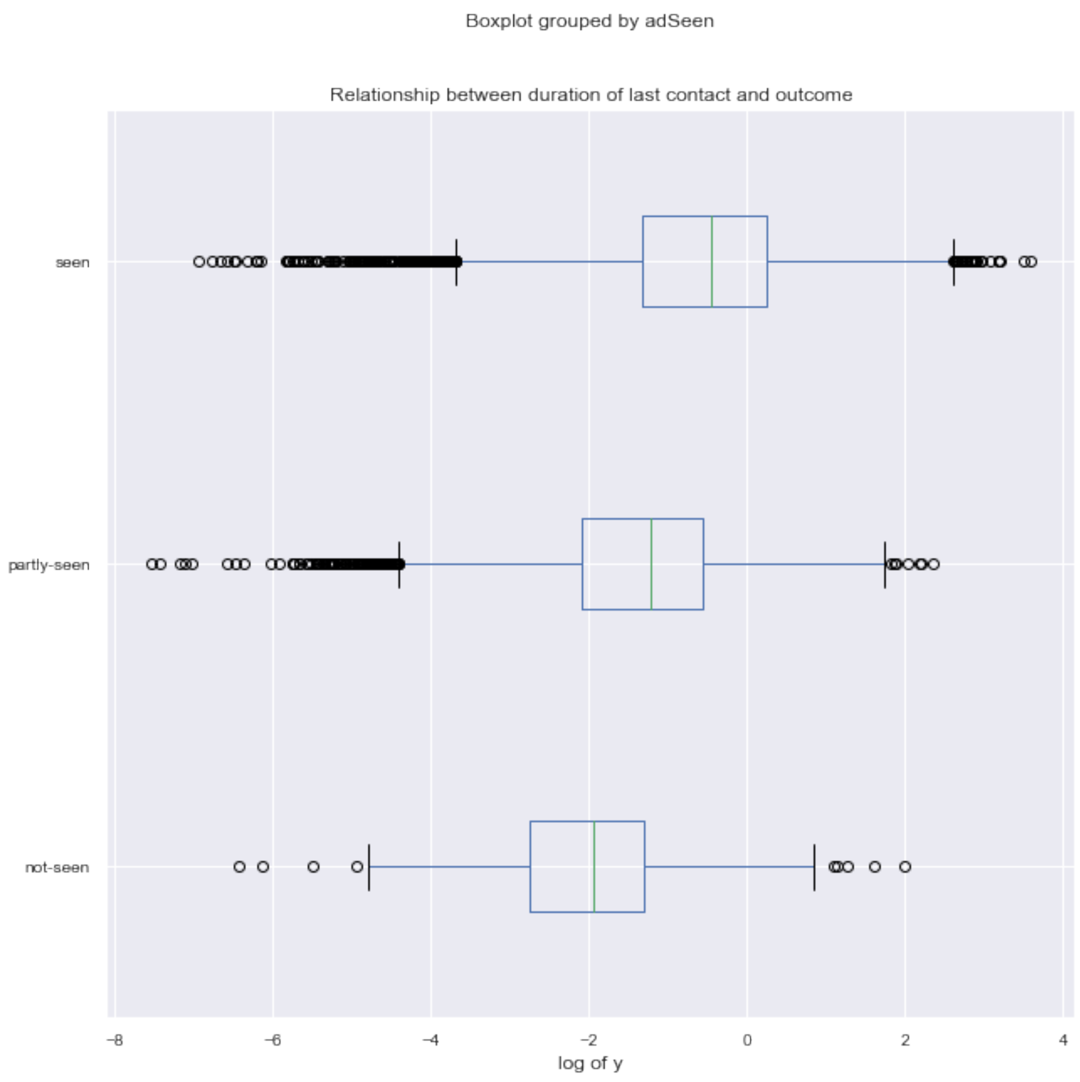
Device code 1 showed the lowest (log of) y values and device code 3 perform the best and also is the most consistent one.

```
In [55]: ad_removed.dropna().boxplot(column='log_y',by='dow',vert=False, figsize=(10,10),fonts
         plt.xlabel('Duration')
         plt.ylabel(' ')
         plt.title('Relationship between duration of last contact and outcome')
         plt.show()
```



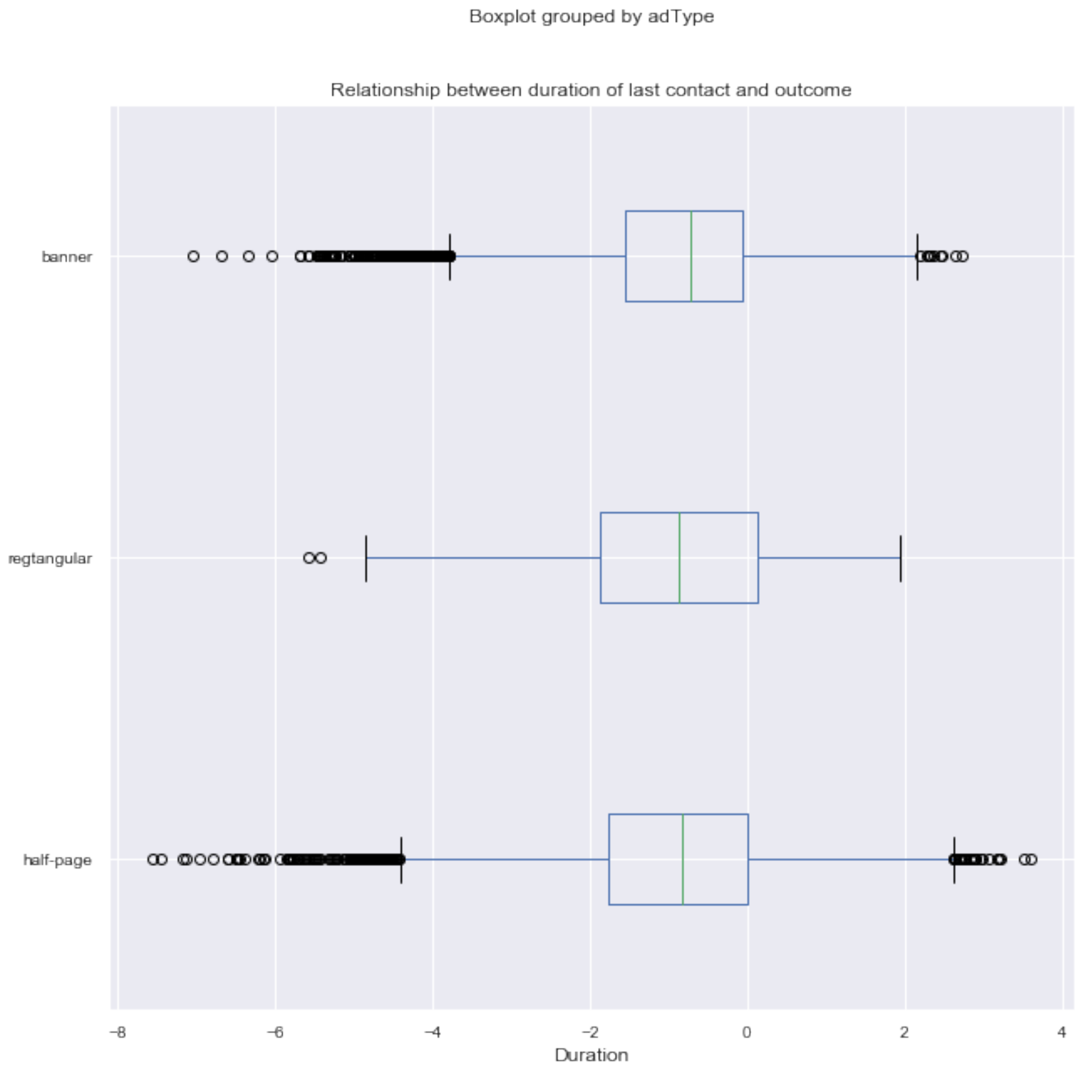
The day of the week variable does not show much differnces among the options, we'll drop this variable from the modelling process

```
In [56]: ad_removed.dropna().boxplot(column='log_y',by='adSeen',vert=False, figsize=(10,10),fo
         plt.xlabel('log of y')
         plt.ylabel(' ')
         plt.title('Relationship between duration of last contact and outcome')
         plt.show()
```



The new-ly created adSeen variables suggesting that ads that have been seen have higher (log of) y values.

```
In [57]: ad_removed.dropna().boxplot(column='log_y',by='adType',vert=False, figsize=(10,10),for
         plt.xlabel('Duration')
         plt.ylabel(' ')
         plt.title('Relationship between duration of last contact and outcome')
         plt.show()
```



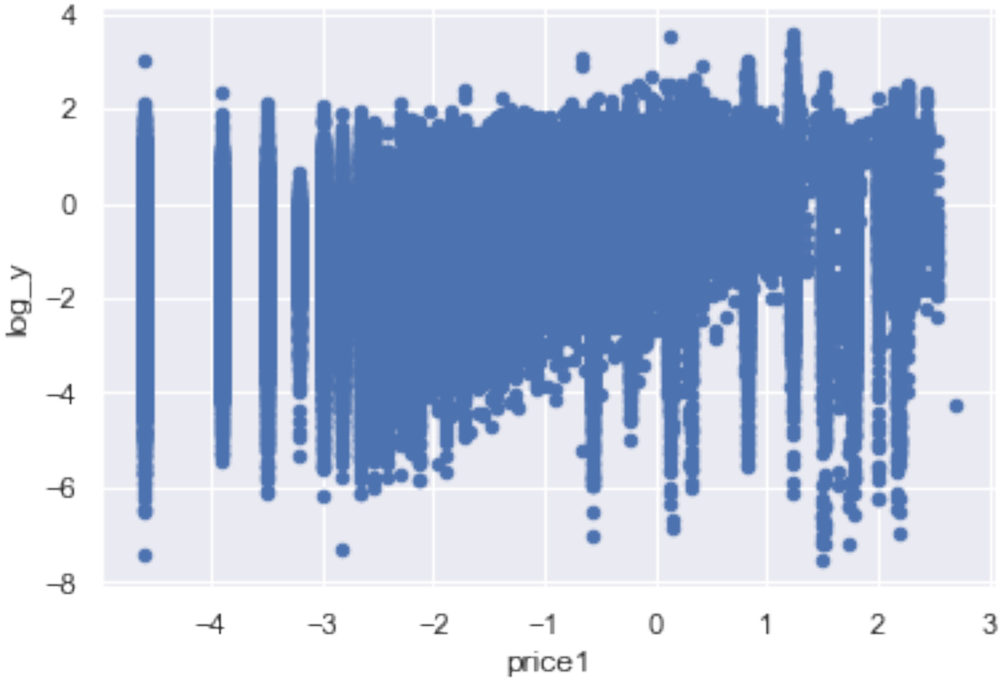
The new adType variable showed that regtangular ads perform more consistent compare to the other two.

3.2.2 Interaction between numeric features and Target Feature

```
In [58]: for col in ['price1', 'price2','price3','ad_area', 'ad_ratio','requests','impression'
             plt.figure(figsize=(6,2))


             ad_removed.plot.scatter(x=col, y='log_y')
             plt.show()
             i = 1 + i
```

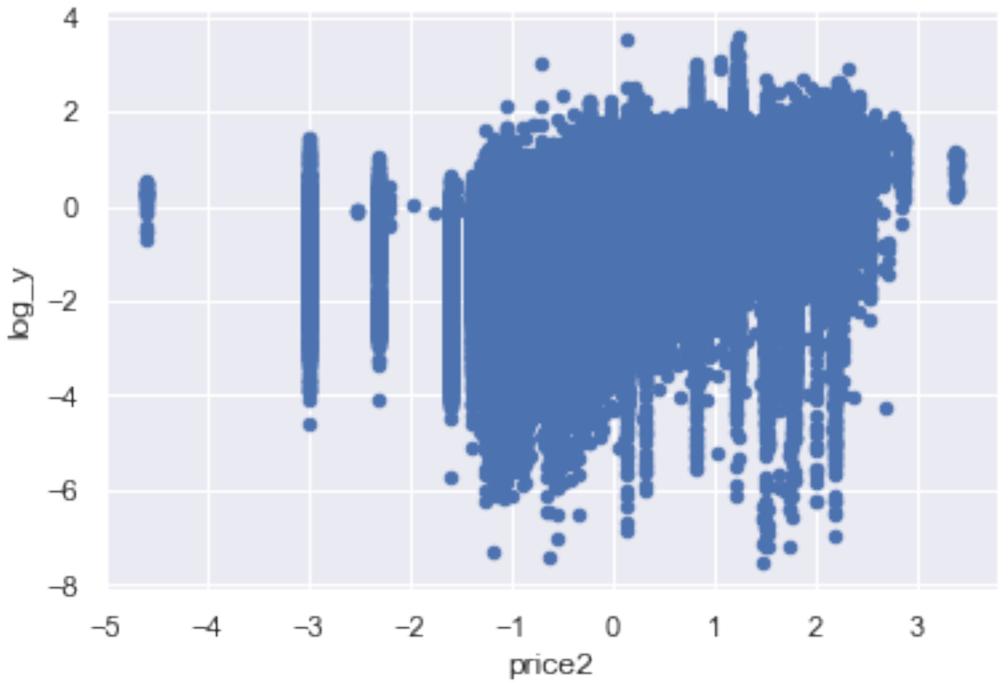
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



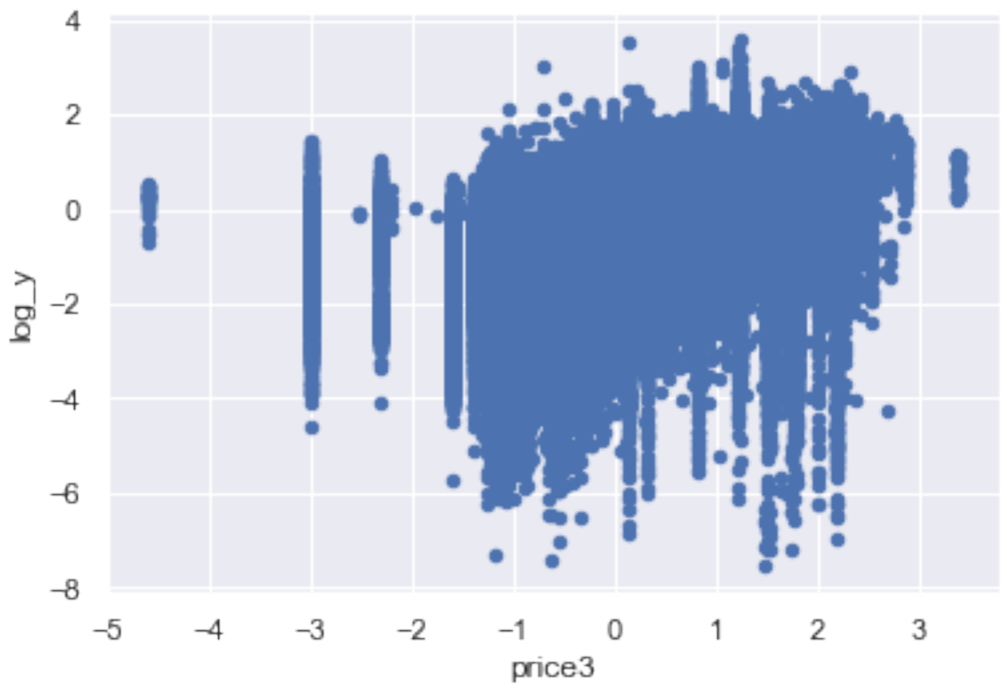
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



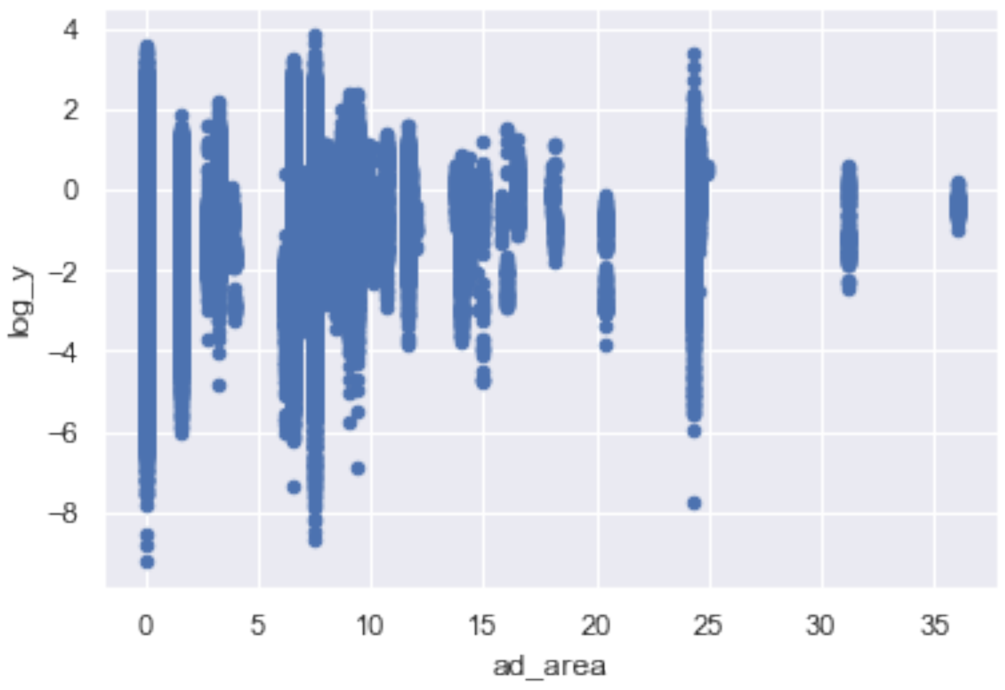
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



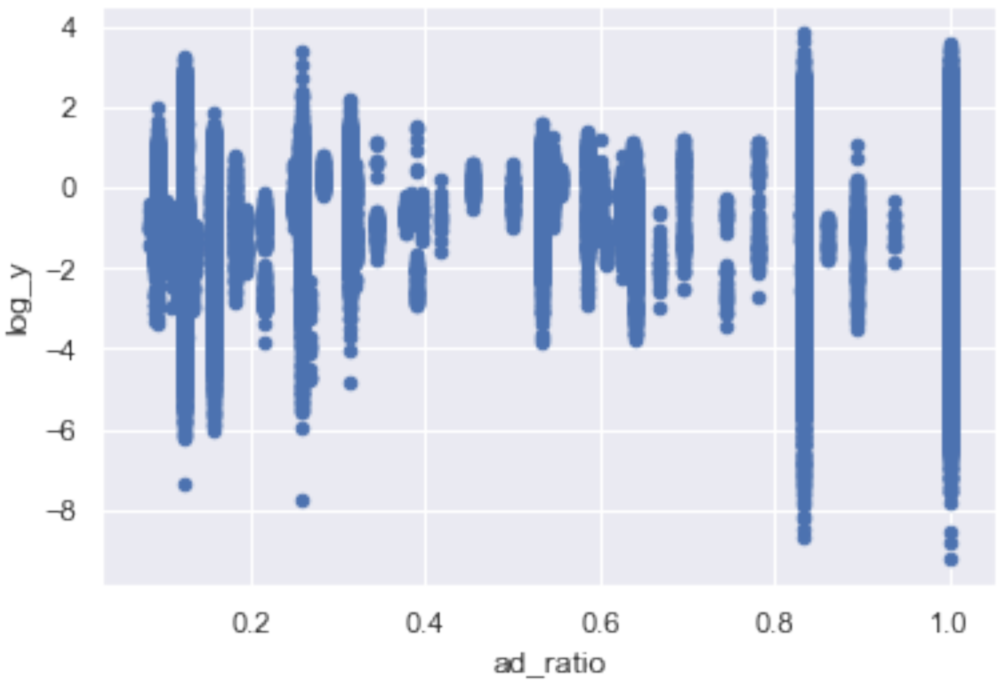
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



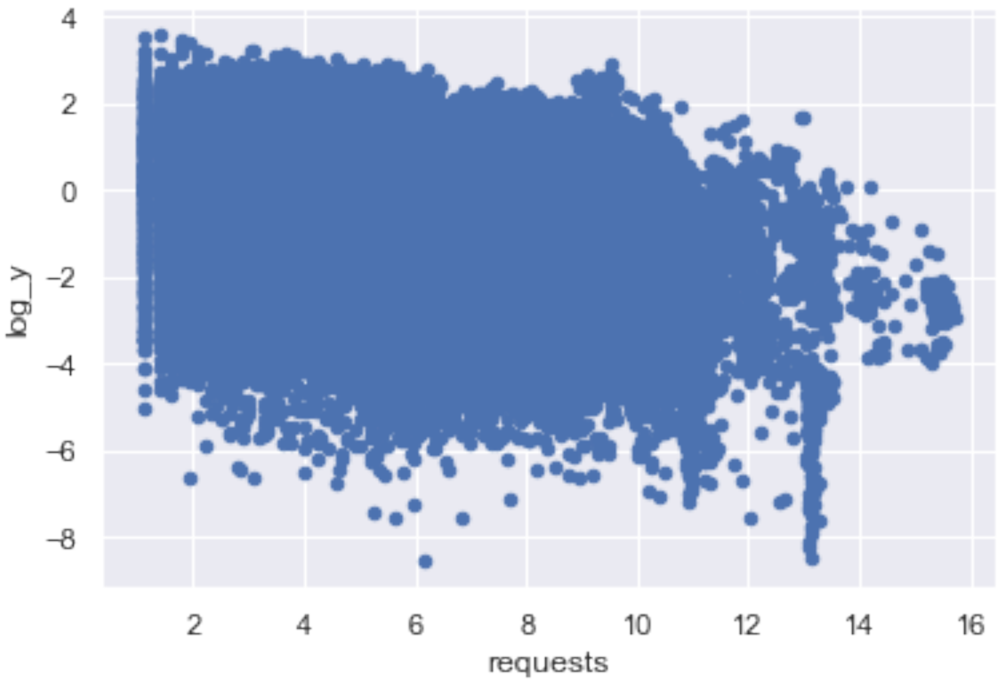
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



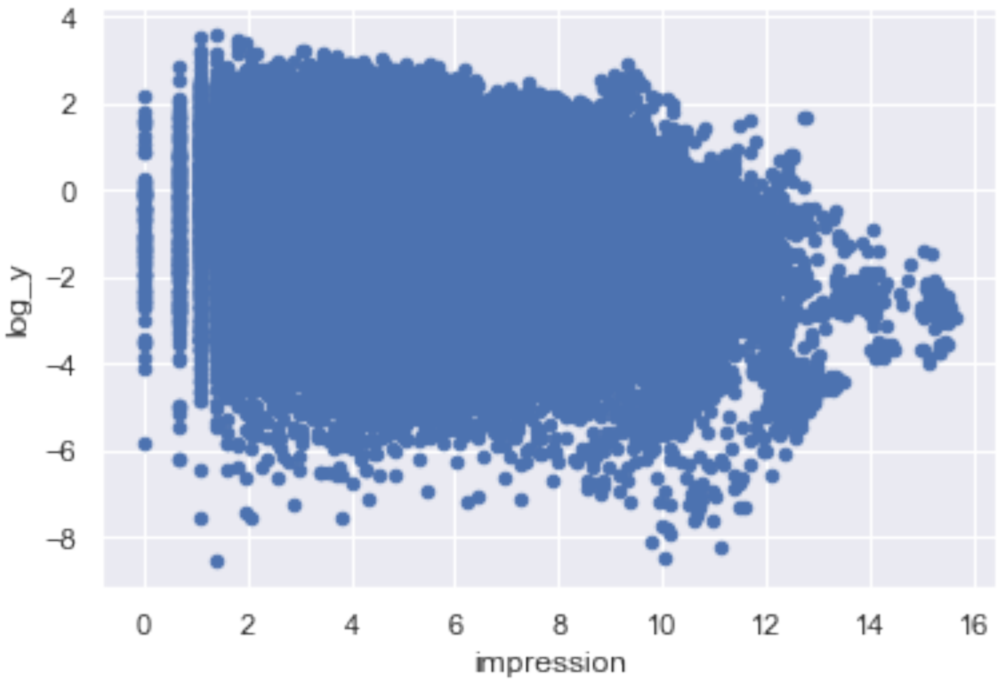
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



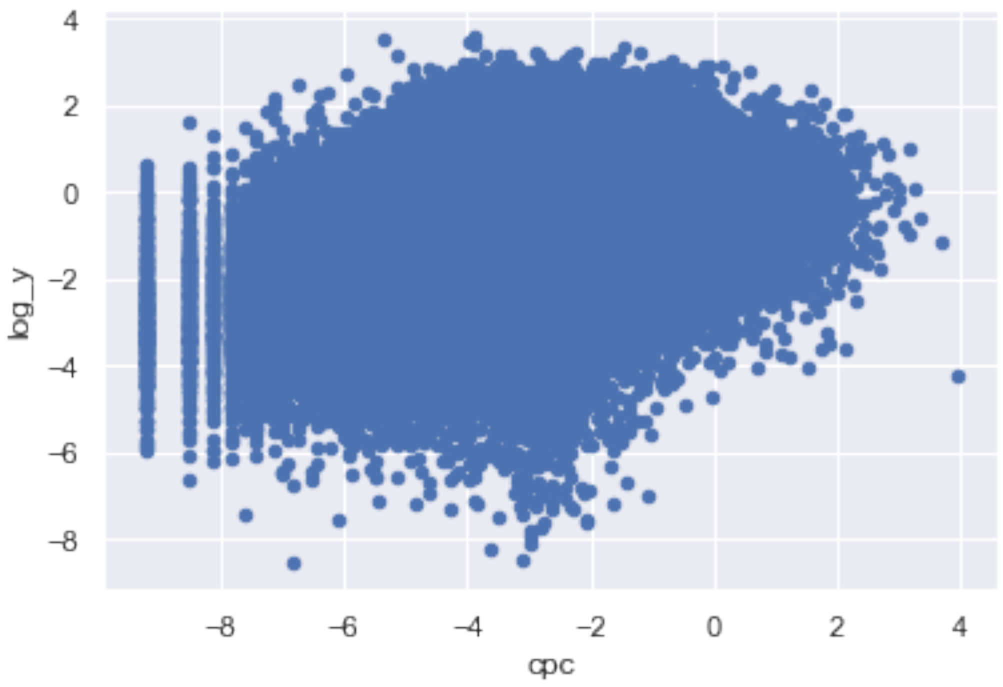
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



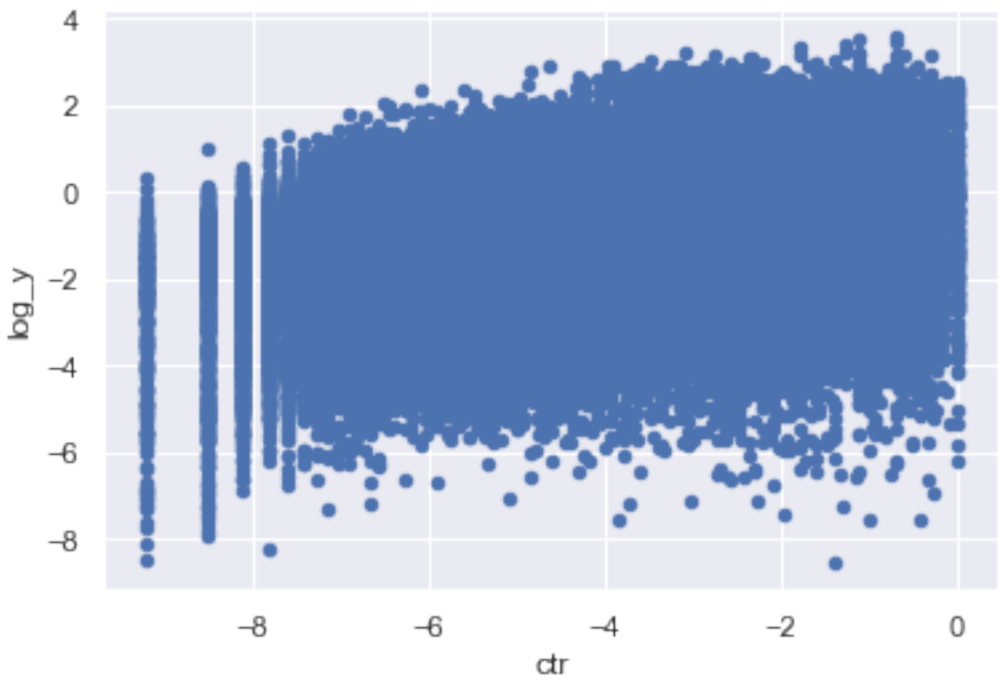
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



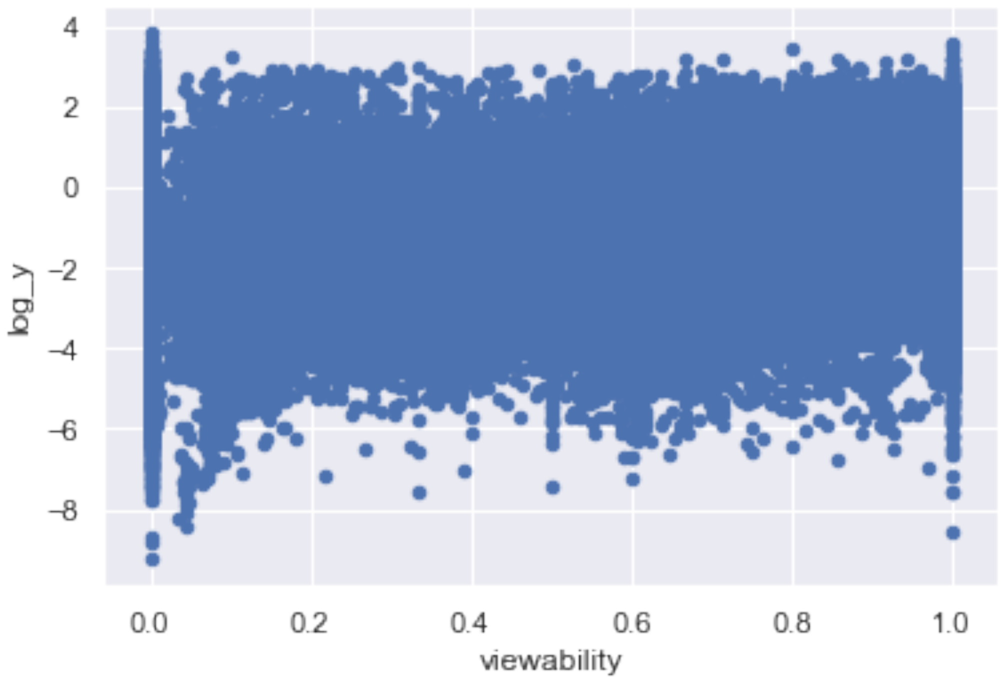
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



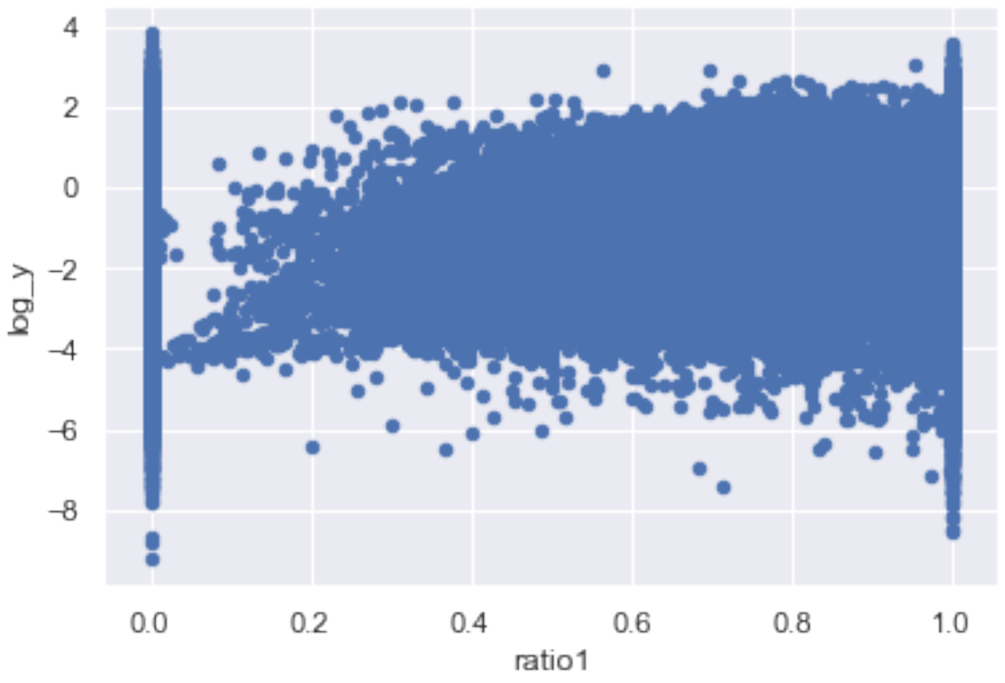
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



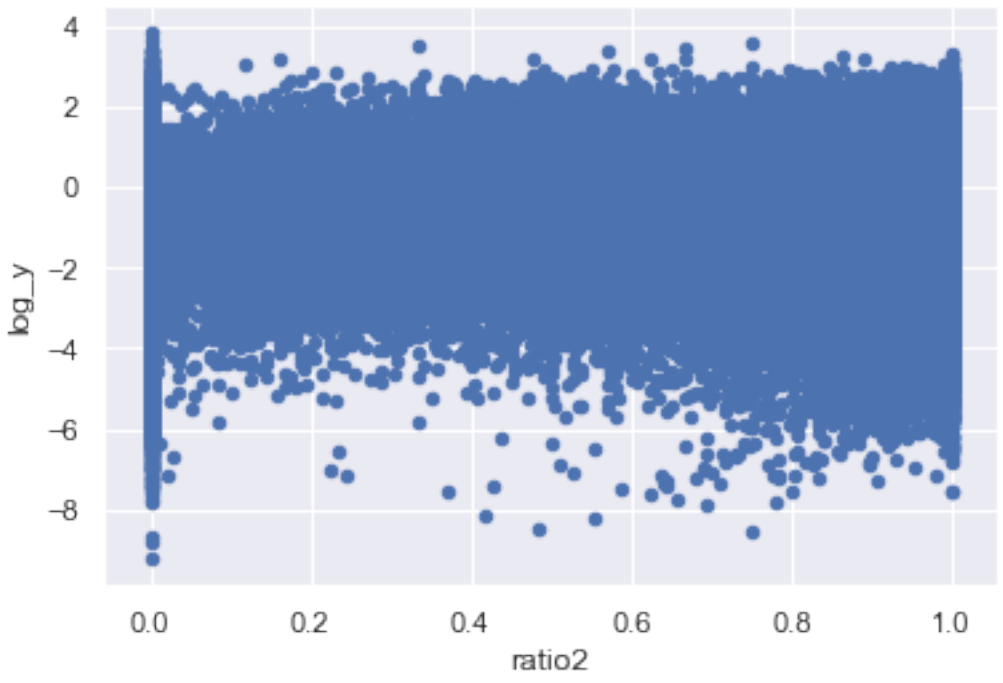
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



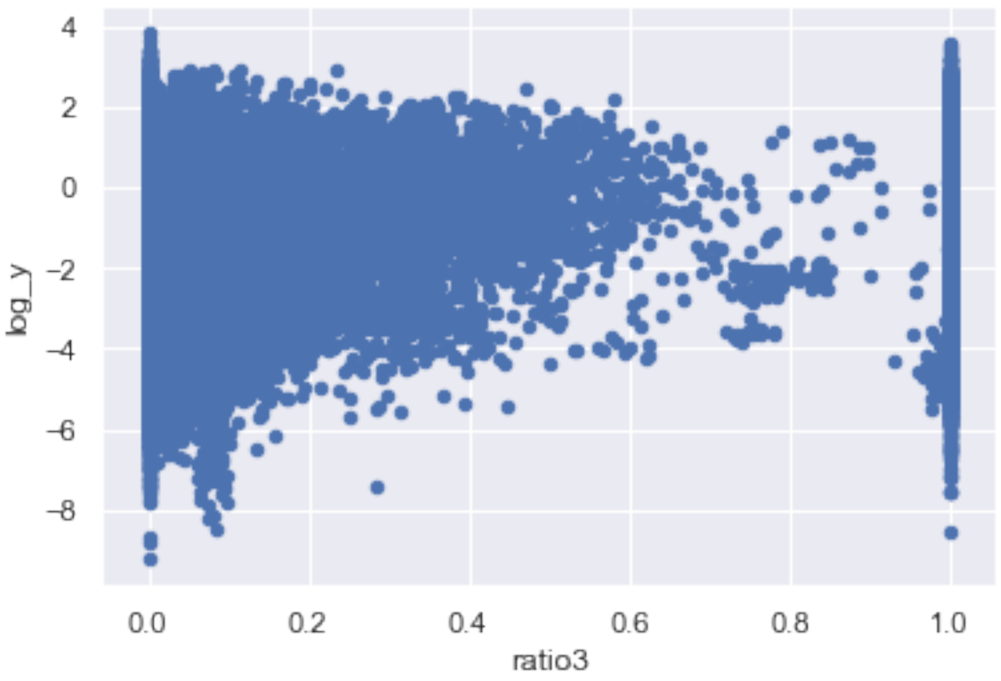
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



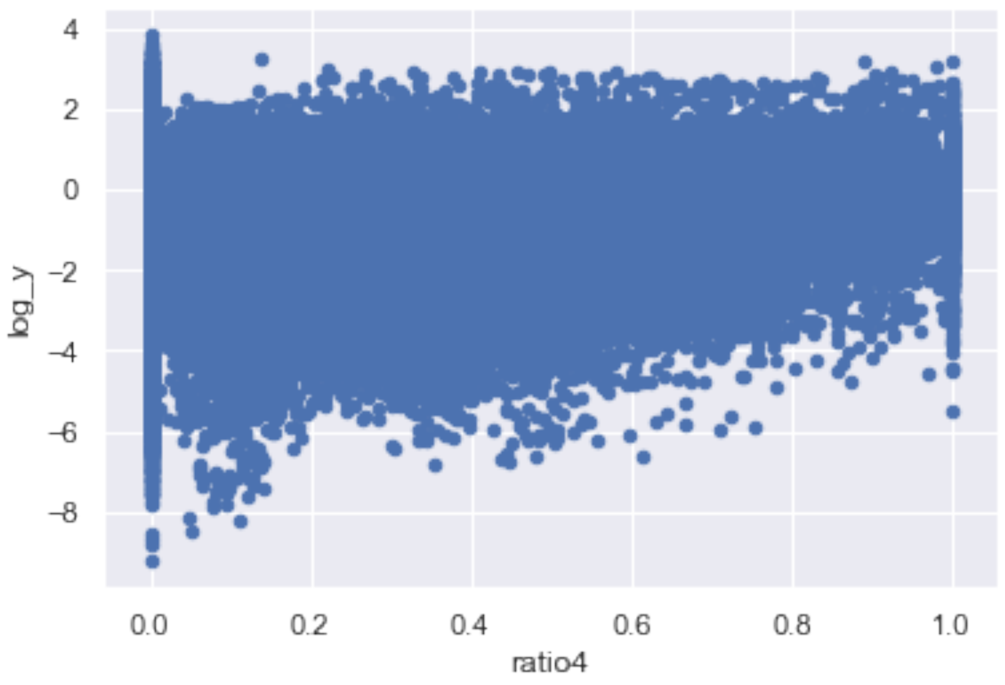
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



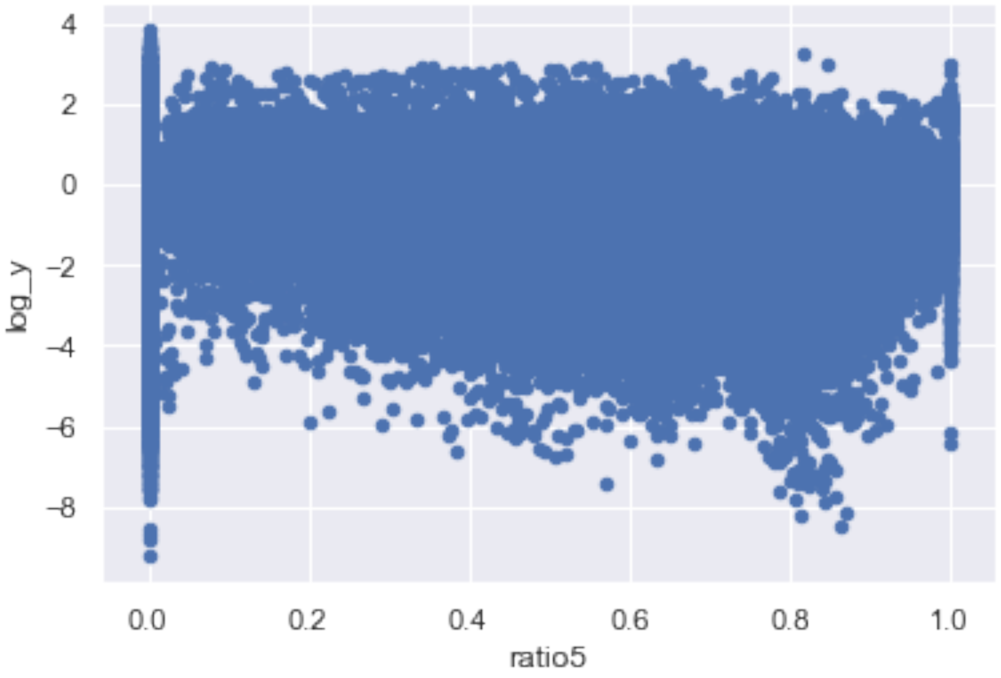
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```

```
<Figure size 432x144 with 0 Axes>
```



There are a few keys things to point out from the series of figures showed about.

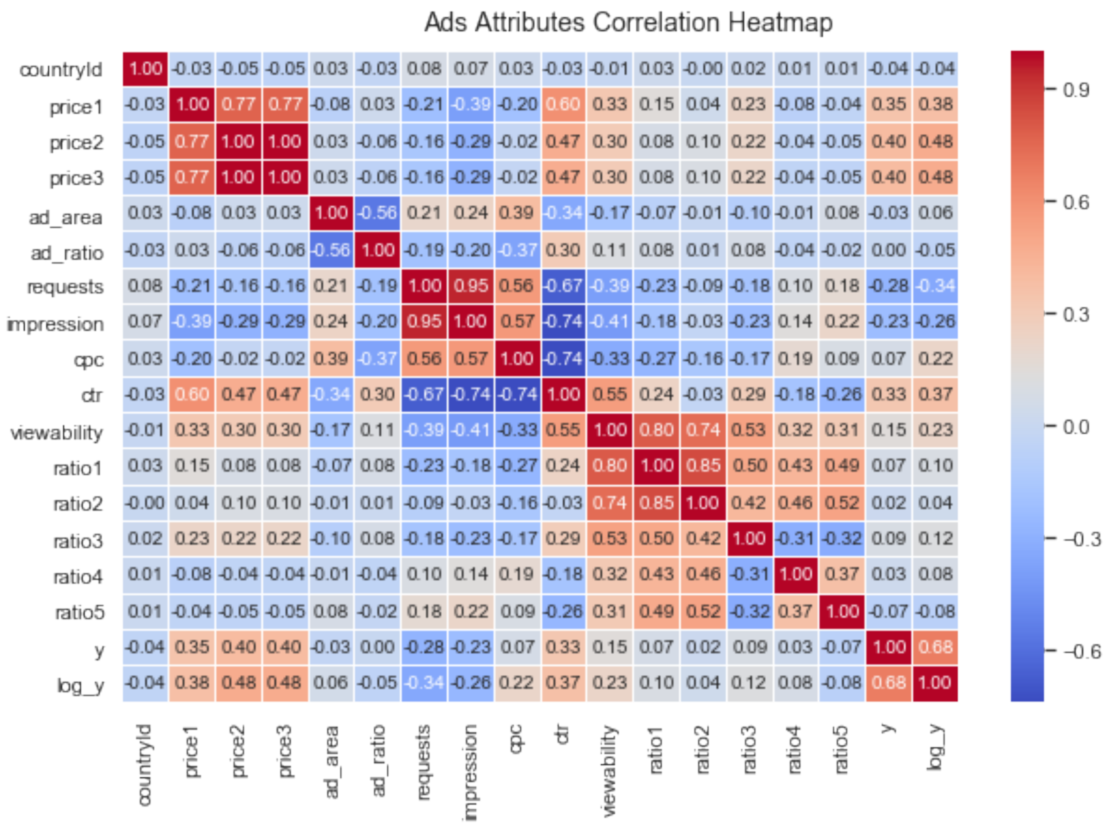
Firstly, all price variables(price1, price2 and price2) showed that the (log of) prices and y values do not suggest any linear relationships. The highest y values seem to appear round the area when (log of) price is around 1.

Secondly, the ad_area variables suggests that there are three different groups, we've already dealt with this previvously. (the logs of) The requests and impression appear to have to negative relationships with the y values. The y values drop as the (logs of) requests and impression decrease.

As the(log of) cpc increases y value increases untill (log of) cpc of around -1.8, then the y values decease at the similar rate.

The (log of) ctr and the y values appear to have some positive linear relationship, as the (log of) ctr increases the y values increases. However the variance of the y values also increase. ratio1 and ratio2 have slight positive linear relationships with y values; while ratio3 has a negative one; ratio4 and ratio5's relationships with y values are unclear.

```
In [59]: # Correlation Matrix Heatmap
         f, ax = plt.subplots(figsize=(10, 6))
         corr = ad_removed.corr()
         hm = sns.heatmap(round(corr,2), annot=True, ax=ax, cmap="coolwarm",fmt='.2f',
                          linewidths=.05)
         f.subplots_adjust(top=0.93)
         t= f.suptitle('Ads Attributes Correlation Heatmap', fontsize=14)
```



The heat map of correlations shows some interesting information, the all the prices and ctr are positive correlated with y values; impression and requests have slight negative correlations with y values.

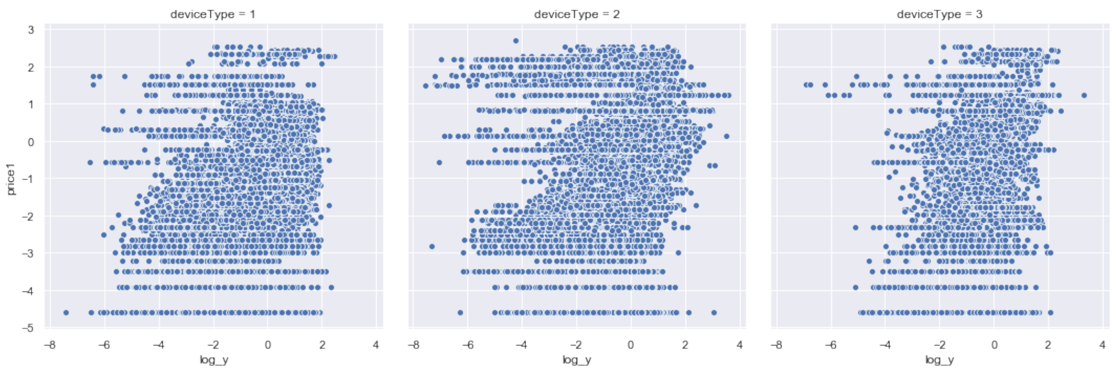
All the prices are highly correlated to each other, we may need to consider a single variable to replace these 3.

ad area and ad_ratio only have correlations with cpc and ctr respectively, however they have little correlations with the y values.

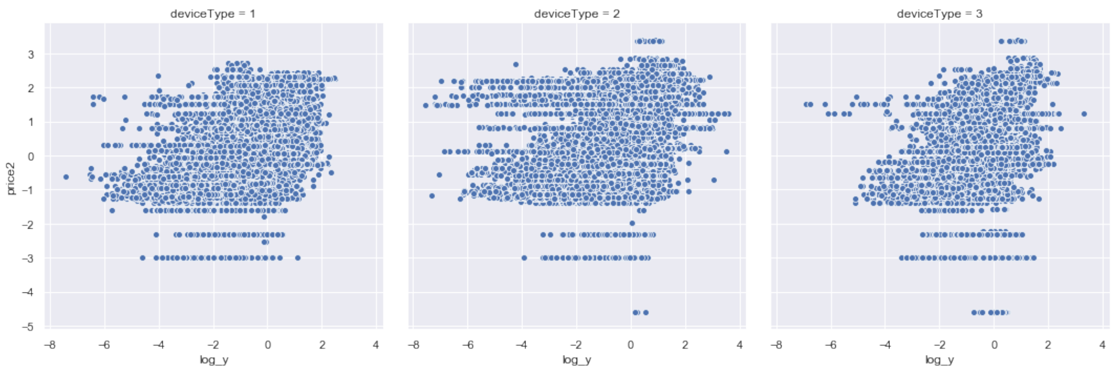
ratio2,coutryId, ratio4 showed very little correlations with the y values. We'll consider dropping these variables for better modelling process. However, it is highly dependent on which machine learning algorithms used.

3.3 Interactions between categorical variables, numeric variables and the target variables

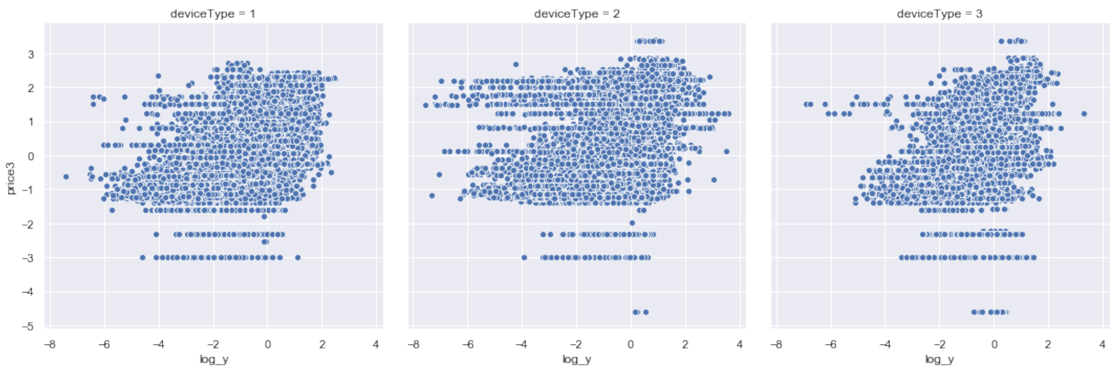
```
In [60]: sns.relplot(x="log_y", y="price1", col="deviceType", data=ad_removed);
```



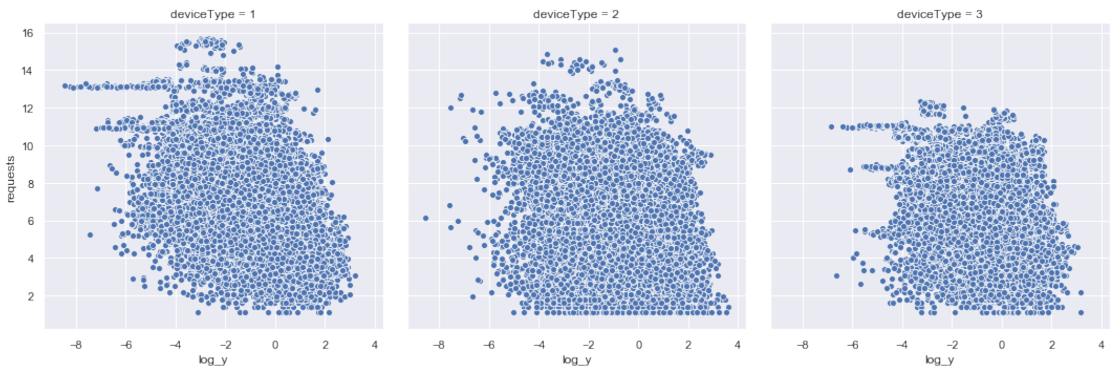
```
In [61]: sns.relplot(x="log_y", y="price2", col="deviceType", data=ad_removed);
```



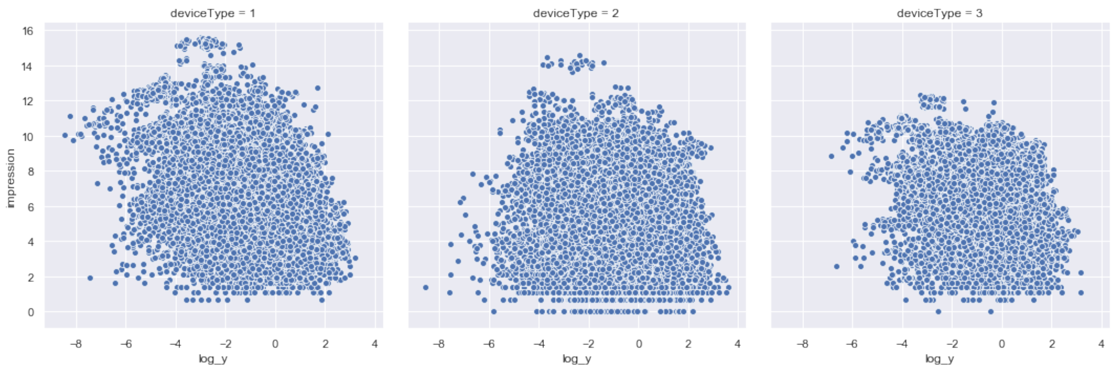
```
In [62]: sns.relplot(x="log_y", y="price3", col="deviceType", data=ad_removed);
```



```
In [63]: sns.relplot(x="log_y", y="requests", col="deviceType", data=ad_removed);
```

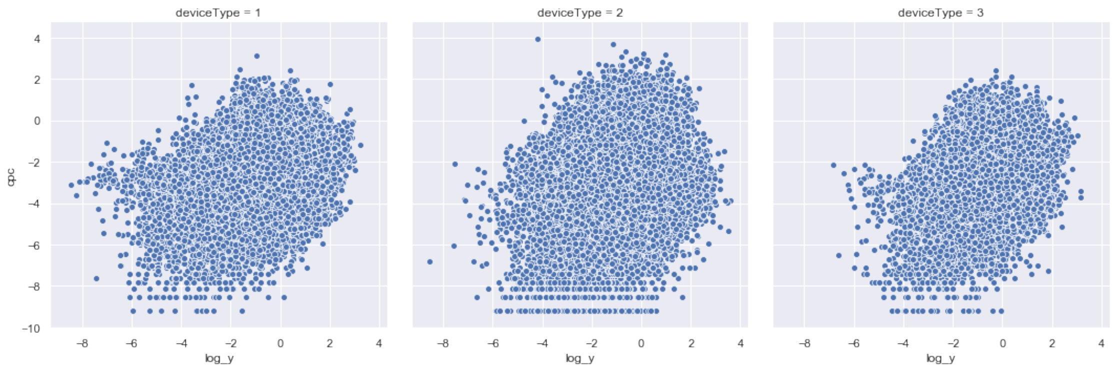


```
In [64]: sns.relplot(x="log_y", y="impression", col="deviceType", data=ad_removed);
```

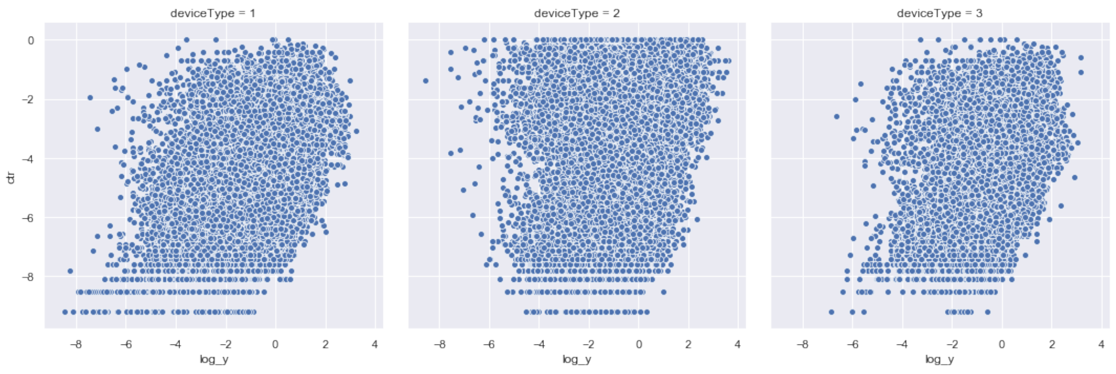


We can observe the negative correlations between requests and (log of) y is stronger in device 1 and 2. In addition, the negative correlation between impressions and (log of) y is clearer in device 1.

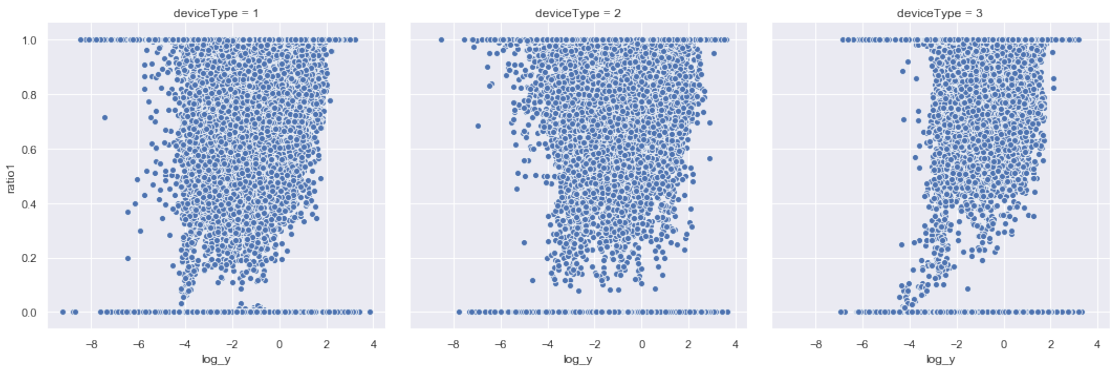
```
In [65]: sns.relplot(x="log_y", y="cpc", col="deviceType", data=ad_removed);
```



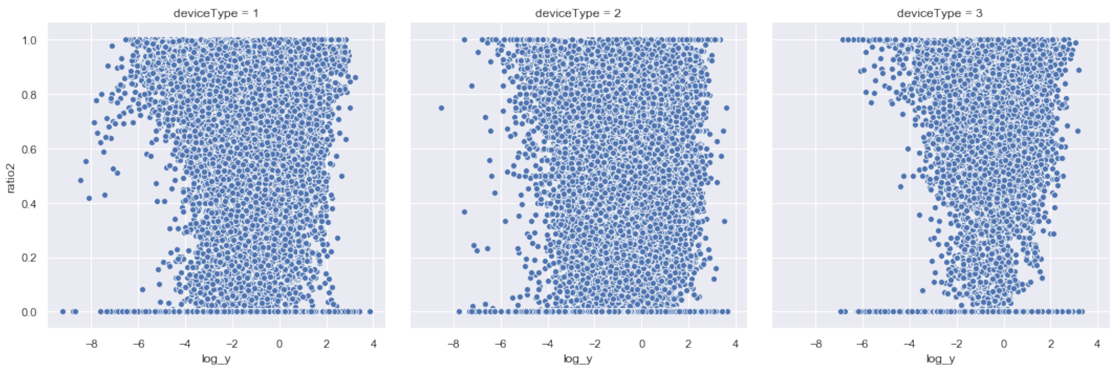
```
In [66]: sns.relplot(x="log_y", y="ctr", col="deviceType", data=ad_removed);
```



```
In [67]: sns.relplot(x="log_y", y="ratio1", col="deviceType", data=ad_removed);
```

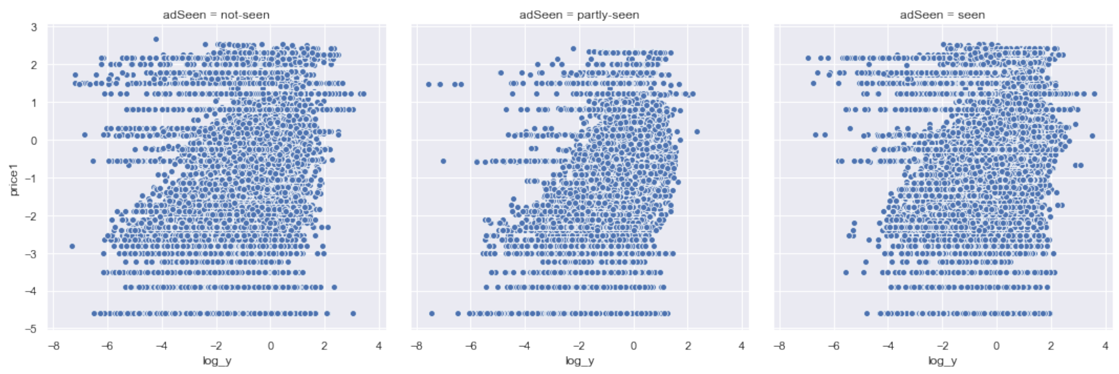


```
In [68]: sns.relplot(x="log_y", y="ratio2", col="deviceType", data=ad_removed);
```

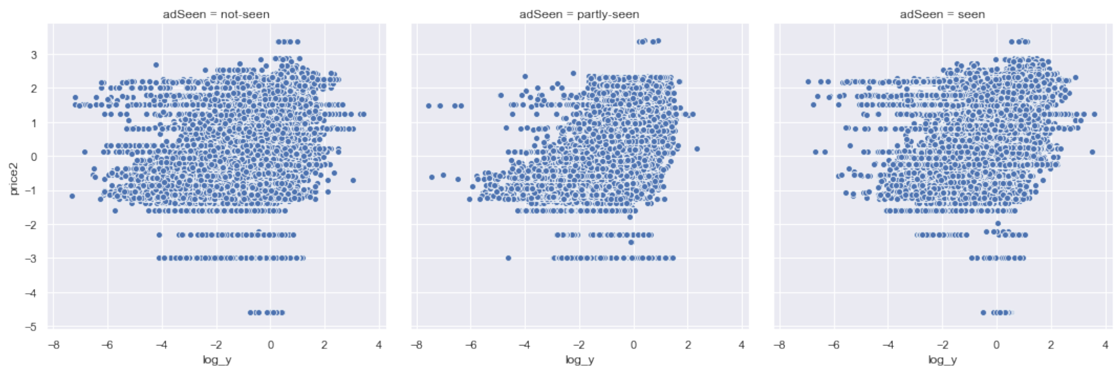


The positive correlations between ctr and (log of) y and cpc and (log of) y can be seen quite clearly in all the devices. However, the variance of these relationships appear to be larger in device 1 and 2.

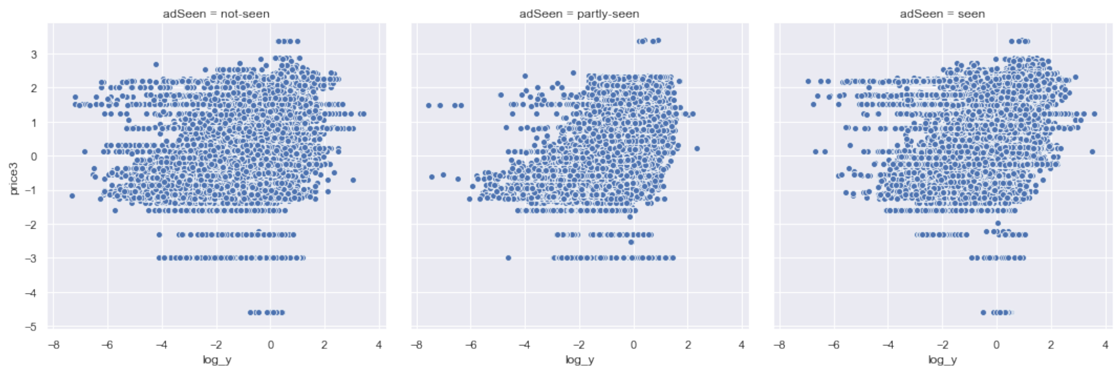
```
In [69]: sns.relplot(x="log_y", y="price1", col="adSeen", data=ad_removed);
```



```
In [70]: sns.relplot(x="log_y", y="price2", col="adSeen", data=ad_removed);
```

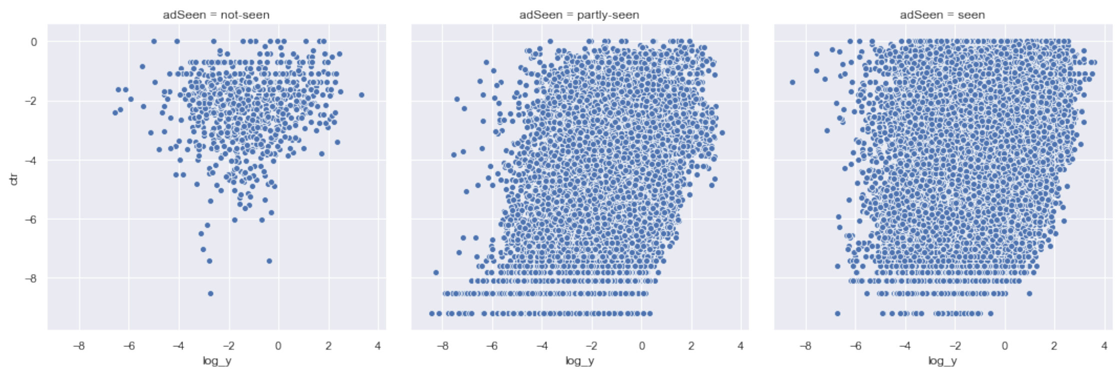


```
In [71]: sns.relplot(x="log_y", y="price3", col="adSeen", data=ad_removed);
```

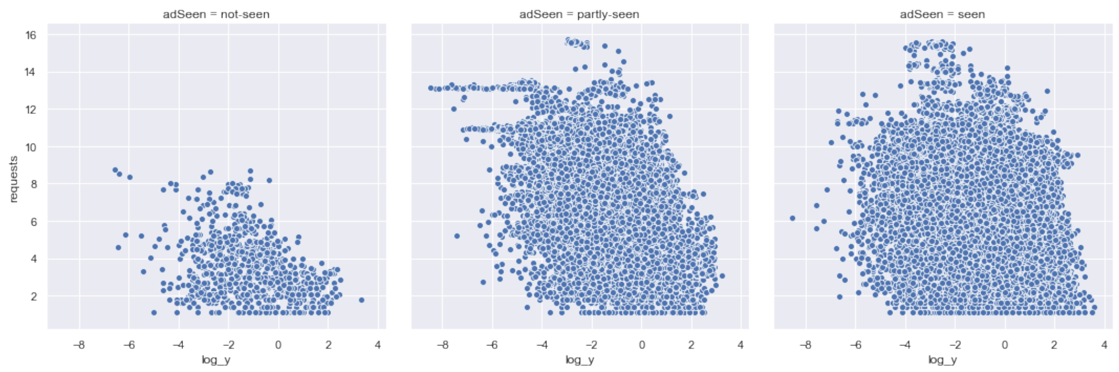


There is not much different between the group, this could be the results of the partions range; further considerations should be taken during the modelling process.

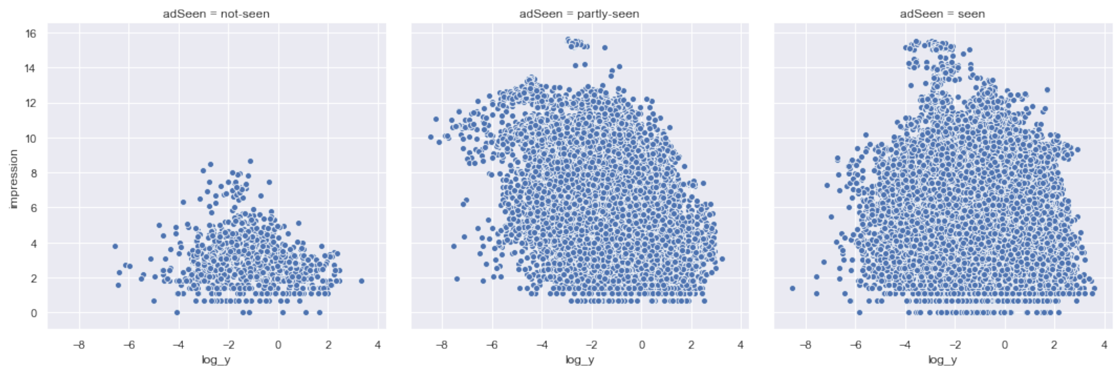
```
In [72]: sns.relplot(x="log_y", y="ctr", col="adSeen", data=ad_removed);
```



```
In [73]: sns.relplot(x="log_y", y="requests", col="adSeen", data=ad_removed);
```

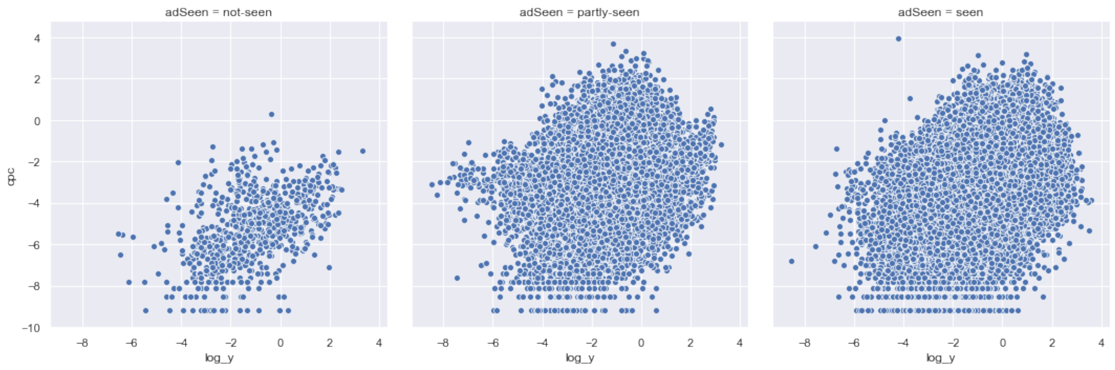


```
In [74]: sns.relplot(x="log_y", y="impression", col="adSeen", data=ad_removed);
```

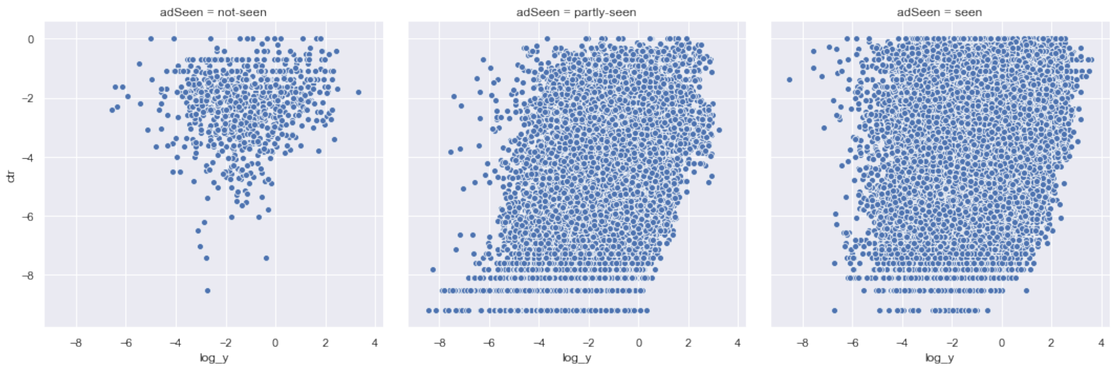


Once again, the negative correlations between impressions and requests with y exhibits clearly in all groups. Partly-seen ads seem to have smaller varince

```
In [75]: sns.relplot(x="log_y", y="cpc", col="adSeen", data=ad_removed);
```

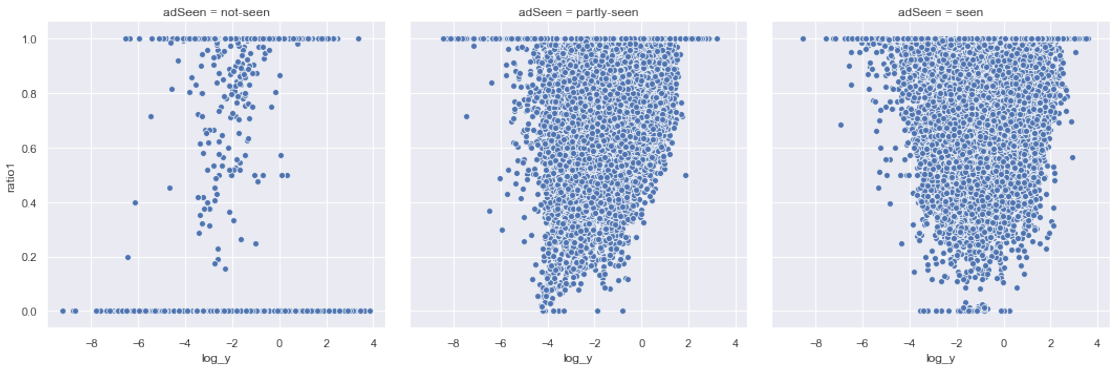


```
In [76]: sns.relplot(x="log_y", y="ctr", col="adSeen", data=ad_removed);
```

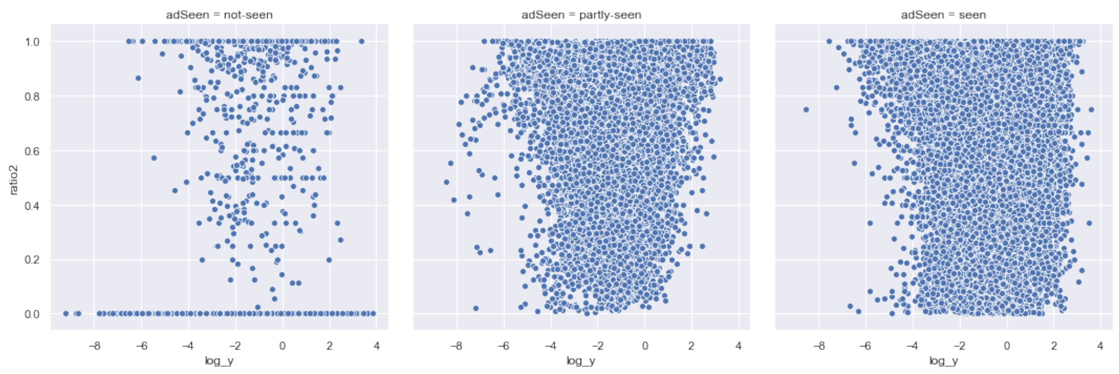


Ads that has been seen appears to cost more; the positive correlation between cpc and (log of) y is very clear here. Besides, ads that are partly seen showed smaller variance in ctr.

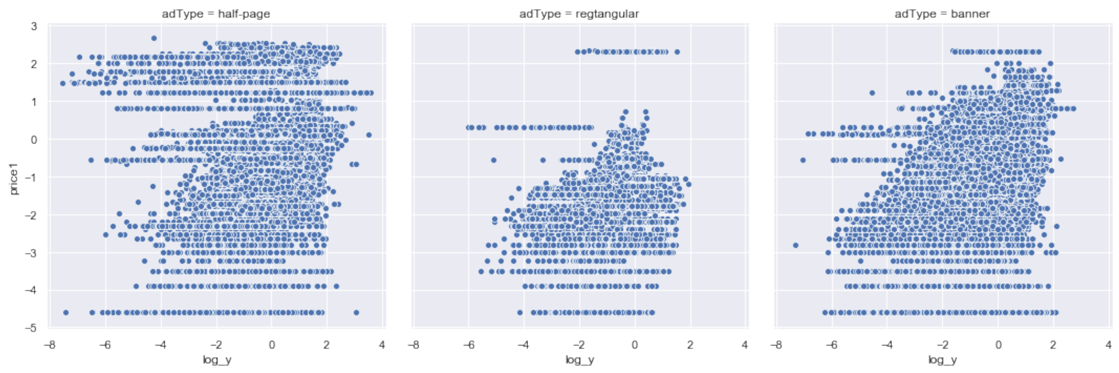
```
In [77]: sns.relplot(x="log_y", y="ratio1", col="adSeen", data=ad_removed);
```



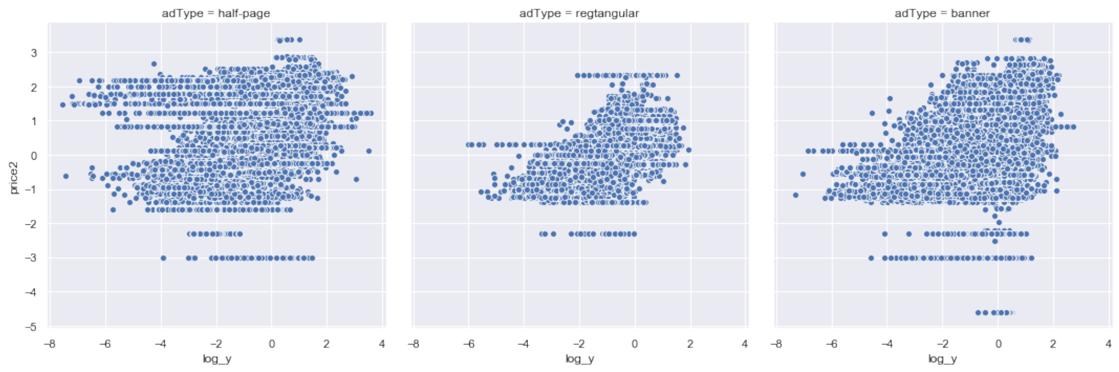
```
In [78]: sns.relplot(x="log_y", y="ratio2", col="adSeen", data=ad_removed);
```



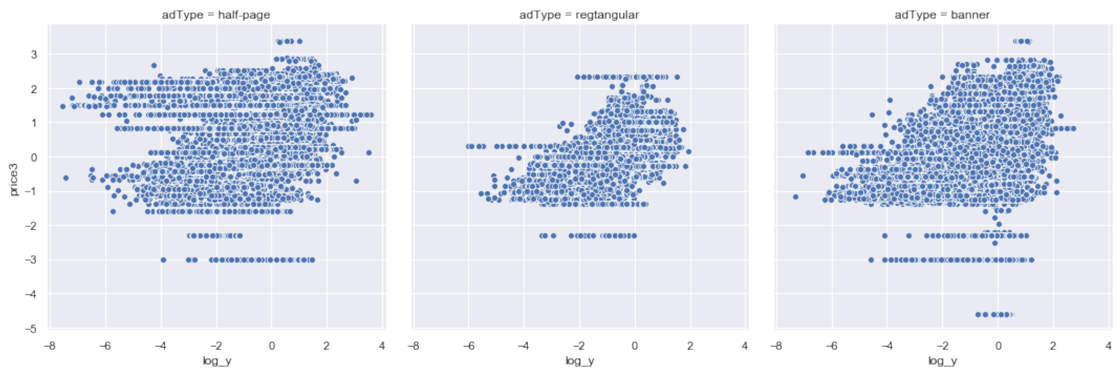
```
In [79]: sns.relplot(x="log_y", y="price1", col="adType", data=ad_removed);
```



```
In [80]: sns.relplot(x="log_y", y="price2", col="adType", data=ad_removed);
```

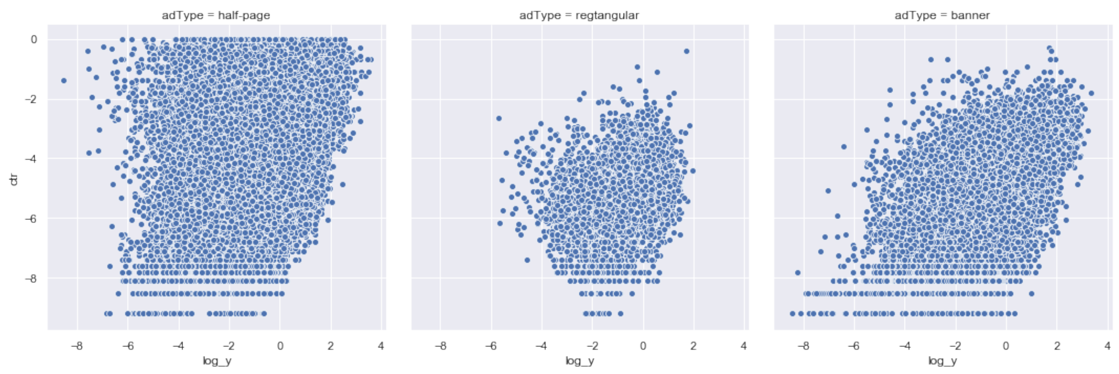


```
In [81]: sns.relplot(x="log_y", y="price3", col="adType", data=ad_removed);
```



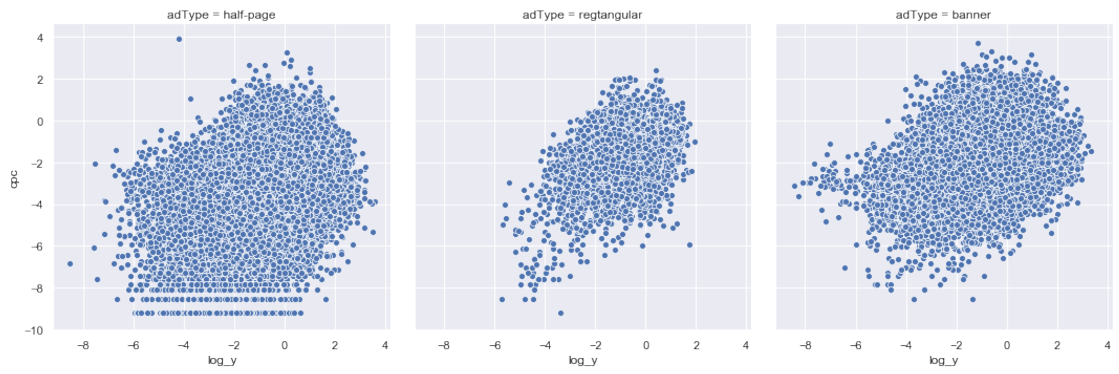
The postive correlations are clear among all the the types of ads; regtangular ads showed smallest variance.

```
In [82]: sns.relplot(x="log_y", y="ctr", col="adType", data=ad_removed);
```



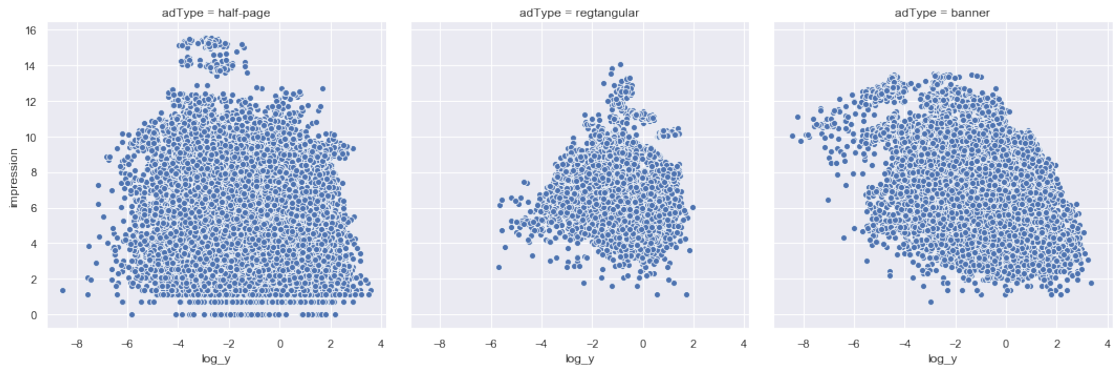
Banner ads appears to perform best in click through rate.

```
In [83]: sns.relplot(x="log_y", y="cpc", col="adType", data=ad_removed);
```

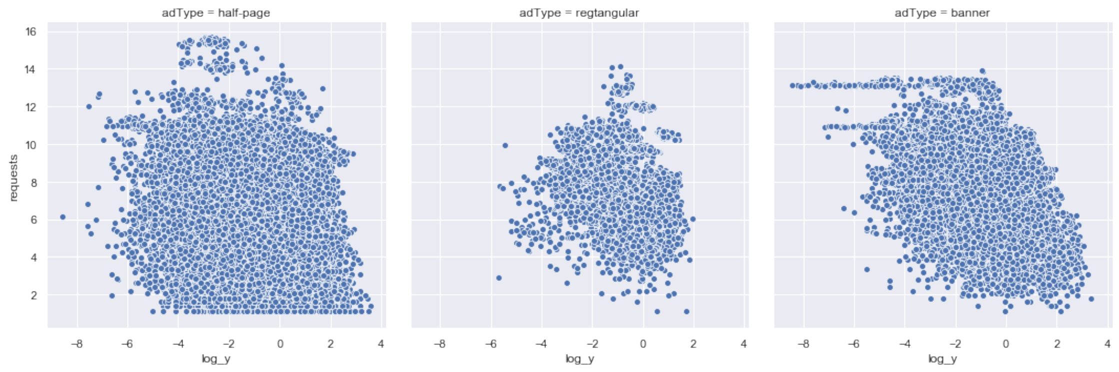


Regtangular ads appears to be the most cosistent in term in performance , where the correlation between cpc and (log of) y in half-page ads exhibits largest variance comparing to the other types of ads.

```
In [84]: sns.relplot(x="log_y", y="impression", col="adType", data=ad_removed);
```



```
In [85]: sns.relplot(x="log_y", y="requests", col="adType", data=ad_removed);
```



In general, all the correlations between (log of) y and the numeric variables appear to be more obvious in some categories than others. The functionalities of the ratio-variables are not very clear and they did not showed much correlations with the target variables. In additions, the ratio-variables represent information regrading to the metrics; we will consider leave these ratios out when using explainable algorithms such as decision tree.

4 Summary

In Phase 1, we have done a few manipulations to the data. We have made some educated guesses and assumptions based on our own knowledge regarding Google Ads, however, proper expert domain knowledge will improve the processing data tremendously.

Firstly, the data has been cleaned, zero values in some of the variables were replaced with NA for better visualisation. An additional step of marking them using a constant will be considered in the next phase depends on which machine learning algorithm we use.

Secondly, transformations were done to some numeric variables including price1, price2, price3, cpc, ctr and the target variables. Some of them displays normal distributions, which will be beneficial for modelling process.

Thirdly, a portion of the data was left out for various of reason. We also construct 2 new categorical variables adSeen and asType.

From the data exploration, we found that deviceType, adSeen,adType, price1,price2, price3, requests,impression,cpc,ctr, viewability, ratio1 and ratio3 were potentially useful features in predicting the income classes.