MATH2319 Machine Learning

Phase 1 - Assignment 1 - Semester 1, 2019

Ash Olney s3686808 Phil Steinke s3725547

Objective

The Google advertising data provided for the Machine Learning Kaggle Competition will be preprocessed for machine learning.

Setup

In [11]:

```
import pandas as pd
import numpy as np
import altair as alt
import math
from scipy import stats
from sklearn import preprocessing
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
import os
import io
```

In [2]:

```
__file_name__ = 'advertising_train.csv'
data = pd.read_csv(__file_name__)
```

In [3]:

```
# consistent naming of columns (minus camelCase):
labelNames = ['case_id','company_id', 'country_id', 'device_type', 'day', 'dow',
'price1', 'price2', 'price3', 'ad_area', 'ad_ratio', 'requests', 'impression',
'cpc', 'ctr', 'viewability', 'ratio1', 'ratio2', 'ratio3', 'ratio4', 'ratio5',
'y',]
data.columns = labelNames
```

The Data

Features in this data set are as follows:

Feature Name	Data Type	Description	Transform	# DataVis?
company_id	categorical	Company ID of record	oneHotEncode() #	
country_id	categorical	Country ID of record *iso3166 1.0*	oneHotEncode()? binary/scaling it	*iso3166 1.0* to # country
device_type	categorical	Device type of record	oneHotEncode()	1 Desktop, 2 mobile, 3 tablet, 5 # ConnectedTv
* **day**	integer	Day of record - between 1 (oldest) and 30 for train, - between 31 (oldest) and 35 for test,	TODO: add a global variable for whether it's training/test #	
* **dow**	categorical	Day of week of the record	- oneHotEncode? dow_friday dow_monday dow_saturday dow_sunday dow_thursday dow_tuesday dow_wednesday - maybe: split into weekday and weekend?	
price1, price2, price3	numeric	Price combination for the record set by the company	#	
ad_area	numeric	area of advertisement (normalized between 0 and 1)	#	
ad_ratio	numeric	ratio of advertisement's length to its width (normalized between 0 and 1)	#	
requests	numeric	Was a request sent to the server to view the ad?	#	
impression	numeric	Is the Google Advertisement viewed?	#	
cpc	numeric	Cost Per Click	#	
ctr	numeric	Click Through Rate - do users click on the advert?	#	
viewability	numeric	[Viewability - metrics for viewable ads (https://support.google.com/google-ads/answer/7029393)](https://support.google.com/google-ads/answer/7029393) A display ad is counted as viewable when at least 50% of its area is visible on the screen for at least 1 second	#	
ratio1, ..., ratio5	numeric	Ratio characteristics related to the record (normalized between 0 and 1)	#	
y (target feature)	numeric	revenue-related metric	- split into separate dataset - don't normalise #	

Note: `ad_area` and `ad_ratio` data description states that it is normalised However, it's max of `36` and `5` means this is a mistake. `ratio1` ... `ratio5` apear to be between 0 to 1. However, because they are ratios, they may not be normalised, and we have normalised the data anyway (because "ratio" implies that it may be between 0-1)

Note: normally we would have [transformed `dow` data as time series (https://datascience.stackexchange.com/questions/17759/# encoding-features-like-month-and-hour-as-categorial-or-numeric)](https://datascience.stackexchange.com/questions/17759/#encoding-features-like-month-and-hour-as-categorial-or-numeric). However, the assignment specifies not to do this.

In [4]:

```
# Shape:
print(data.shape)
```

```
(214128, 22)
```

In [5]:

```
# Dataset datatypes:
print(data.dtypes)
```

```
case_id          int64
company_id       int64
country_id       int64
device_type      int64
day              int64
dow             object
price1         float64
price2         float64
price3         float64
ad_area        float64
ad_ratio       float64
requests         int64
impression       int64
cpc            float64
ctr            float64
viewability    float64
ratio1         float64
ratio2         float64
ratio3         float64
ratio4         float64
ratio5         float64
y              float64
dtype: object
```

In [6]:

```
# Common functions for reuse throughout code:
numeric = list(data.select_dtypes(exclude=['object']).columns)
numericColumnsList = list(data[(numeric)].columns.values)
objects = list(data.select_dtypes(include=['object']).columns)
```

In [7]:

```
# Numerical Column Upper and Lower limits
pd.concat(
    [data[(numeric)].max().round(),
    data[(numeric)].min().round()],
    keys=['Max', 'Min'],
    axis=1)
# This is used to show the data was not normalised, as described
```

Out[7]:

	Max	Min
case_id	214128.0	1.0
company_id	159.0	40.0
country_id	251.0	1.0
device_type	5.0	1.0
day	30.0	1.0
price1	15.0	0.0
price2	63.0	0.0
price3	79.0	0.0
ad_area	36.0	0.0
ad_ratio	5.0	0.0
requests	6701924.0	0.0
impression	6100324.0	0.0
cpc	133.0	0.0
ctr	2.0	0.0
viewability	7.0	0.0
ratio1	1.0	0.0
ratio2	1.0	0.0
ratio3	2.0	0.0
ratio4	1.0	0.0
ratio5	1.0	0.0
y	47.0	0.0

Data Pre-processing

In [8]:

```
dataOriginal = data.copy()

# convert weekday to lowercase
data['dow'] = data['dow'].str.lower()

# convert weekday object to binary variables
# to get around the issue of creating nonsensical quantitative relationships bet
ween category codes
data = pd.get_dummies(data, columns=['dow'], prefix=['dow'])

print(data.dtypes)
# all data is now converted to numerical
```

```
case_id             int64
company_id          int64
country_id          int64
device_type         int64
day                 int64
price1            float64
price2            float64
price3            float64
ad_area           float64
ad_ratio          float64
requests            int64
impression          int64
cpc               float64
ctr               float64
viewability       float64
ratio1            float64
ratio2            float64
ratio3            float64
ratio4            float64
ratio5            float64
y                 float64
dow_friday          uint8
dow_monday          uint8
dow_saturday        uint8
dow_sunday          uint8
dow_thursday        uint8
dow_tuesday         uint8
dow_wednesday       uint8
dtype: object
```

In [9]:

```
# check for nulls
print(data.isnull().sum())
# total: 0 null variables
```

```
case_id          0
company_id       0
country_id       0
device_type      0
day              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
dow_friday       0
dow_monday       0
dow_saturday     0
dow_sunday       0
dow_thursday     0
dow_tuesday      0
dow_wednesday    0
dtype: int64
```

In [16]:

```
# summary statistics
print(data.describe())
```

```
             case_id     company_id     country_id    device_type  \
count  214128.000000  214128.000000  214128.000000  214128.000000
mean   107064.500000      73.332988     119.496180       1.875612
std     61813.573558      47.817556      76.129206       0.787796
min         1.000000      40.000000       1.000000       1.000000
25%     53532.750000      43.000000      56.000000       1.000000
50%    107064.500000      43.000000     102.000000       2.000000
75%    160596.250000      95.000000     197.000000       2.000000
max    214128.000000     159.000000     251.000000       5.000000

                 day         price1         price2         price3  \
count  214128.000000  214128.000000  214128.000000  214128.000000
mean       15.790522       0.438229       0.630178       0.932436
std         8.385557       1.281403       1.481552       1.839991
min         1.000000       0.000000       0.000000       0.000000
25%         9.000000       0.000000       0.000000       0.000000
50%        16.000000       0.010000       0.090000       0.294800
75%        23.000000       0.190000       0.570000       0.985650
max        30.000000      14.690000      63.120000      78.900000

             ad_area       ad_ratio  ...         ratio4           rati
o5  \
count  214128.000000  214128.000000  ...  214128.000000  214128.0000
00
mean        4.724445       0.923402  ...       0.131008       0.1883
00
std         6.273410       0.482055  ...       0.239758       0.2971
21
min         0.000100       0.083330  ...       0.000000       0.0000
00
25%         0.000100       0.833330  ...       0.000000       0.0000
00
50%         0.000100       1.000000  ...       0.000000       0.0000
00
75%         7.500000       1.000000  ...       0.163600       0.3847
00
max        36.000000       5.000000  ...       1.076900       1.2000
00

                   y     dow_friday     dow_monday   dow_saturday  \
count  214128.000000  214128.000000  214128.000000  214128.000000
mean        0.847004       0.129371       0.139337       0.164388
std         1.390593       0.335611       0.346299       0.370628
min         0.000098       0.000000       0.000000       0.000000
25%         0.150415       0.000000       0.000000       0.000000
50%         0.419000       0.000000       0.000000       0.000000
75%         0.959048       0.000000       0.000000       0.000000
max        47.060000       1.000000       1.000000       1.000000

          dow_sunday   dow_thursday    dow_tuesday  dow_wednesday
count  214128.000000  214128.000000  214128.000000  214128.000000
mean        0.158232       0.137240       0.139664       0.131767
std         0.364960       0.344102       0.346639       0.338238
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

[8 rows x 28 columns]
```

In [13]:

```
#split `y` into separate dataset before normalisation
dataUnscaled = data.drop(columns = 'y')
dataTarget = data['y']
# type(dataTarget) # pandas series
```

Data Normalisation

In [14]:

```
dataNormalised = dataUnscaled.copy() # create copy of dataUnscaled
numericFeaturesToScale = [ 'price1', 'price2', 'price3', 'ad_area', 'ad_ratio',
'requests', 'impression', 'cpc', 'ctr', 'viewability', 'ratio1', 'ratio2', 'rati
o3', 'ratio4', 'ratio5', ]

# apply RobustScaler to normalise data
data_scaler = preprocessing.RobustScaler().fit(dataNormalised[numericFeaturesToS
cale])
# Normalise numerical features, as a pandas Dataframe:
dataNormalised[numericFeaturesToScale] = data_scaler.transform(dataNormalised[nu
mericFeaturesToScale])
```

In [15]:

```
def get_outliers(df):
    absolute_normalized = np.abs(stats.zscore(df))
    return absolute_normalized > 3

def get_length_unique_outliers(df):
    # get boolean array of outliers
    outliers = get_outliers(df)
    outliersIndexList = df[outliers].index.values.astype(int)
    # avoid counting a record with multiple outliers twice
    uniqueOutliersIndexList = np.unique(outliersIndexList)
    print(len(np.unique(outliersIndexList)))
    # return len(np.unique(outliersIndexList))

# length of dataframe
print("length of dataframe:")
len(dataUnscaled)

print("Number of Outliers in original data:")
get_length_unique_outliers(data) # 31258 ~15% outliers in original dataset

print("Number of Outliers Ignoring the Target Variable Y:")
lengthDataOutliers = get_length_unique_outliers(dataUnscaled) # 28167, ~13% outl
iers after we removed target

print("Number of Outliers of Normalised Variables:")
lengthDataNormalisedOutlisers = get_length_unique_outliers(dataNormalised) # 281
67, ~13%
lengthDataOutliers == lengthDataNormalisedOutlisers

# we have ~13% outliers both before and after normalisation.
```

```
length of dataframe:
Number of Outliers in original data:
31258
Number of Outliers Ignoring the Target Variable Y:
28167
Number of Outliers of Normalised Variables:
28167
```

Out[15]:

```
True
```

In [16]:

```
# where are the outliers?
def get_outliers_column_count(df):
    for i in numericFeaturesToScale:
        try:
            print(i)
            get_length_unique_outliers(df[i])
        except:
            print(i)

get_outliers_column_count(dataNormalised)
```

```
price1
6172
price2
4987
price3
5378
ad_area
5512
ad_ratio
1550
requests
815
impression
424
cpc
2220
ctr
5727
viewability
50
ratio1
0
ratio2
0
ratio3
0
ratio4
4173
ratio5
3
```

In [17]:

```
# reduce outliers with natural logarithm
def get_natural_log(arr, cols):
    for i in cols:
        try:
            arr[i] = np.log(arr[i]+10)
        except:
            print(i)
get_natural_log(dataUnscaled, numericFeaturesToScale)

get_length_unique_outliers(dataUnscaled)
# outliers with natural log are reduced to 11.2%
```

```
23942
```

Machine Learning Outlier detection:

Given our 11.2% outliers, applying the following will yeild around 1.9% outliers.

Note: We have visualised the data with the above outliers to show how many outliers the data includes using traditional statistical methods

In [24]:

```
dataWithMachineLearningOutliers = dataNormalised.copy()
clf = IsolationForest(behaviour = 'new', max_samples=100, random_state = 1, cont
amination= 'auto')
preds = clf.fit_predict(dataWithMachineLearningOutliers[numericFeaturesToScale])
preds

isolationForestOutliers = dataWithMachineLearningOutliers[preds == -1]
get_length_unique_outliers(isolationForestOutliers)
```

```
3315
```

Data Visualisation

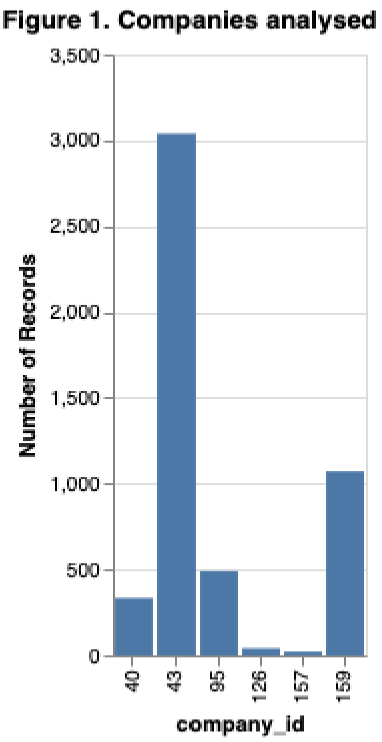
Univariate

In [26]:

```
source = dataNormalised.sample(n=5000)

alt.Chart(source).mark_bar().encode(
    x = "company_id:N",
    y = 'count()',
).properties(
    title='Figure 1. Companies analysed')
```

Out[26]:



Company ID is a nominal variable which will need to be converted into binary variables for phase II. The majority of cases are company 43.

In [27]:

```
alt.Chart(source).mark_bar().encode(
    x = "country_id:N",
    y = 'count()',
).properties(
    title='Figure 2. Records by country')
```

Out[27]:

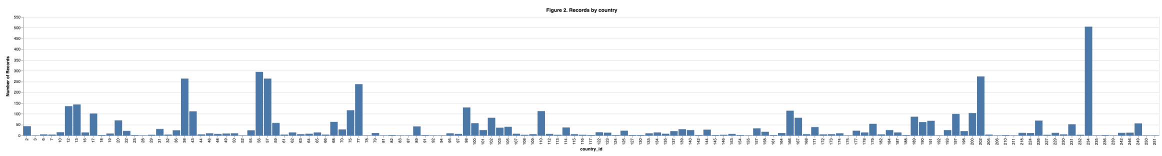
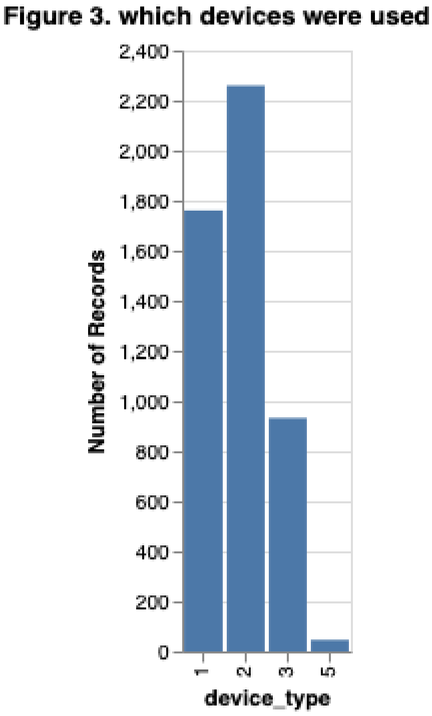


Figure 2 shows that there is a large variation in country of origin. This variable will also need to be converted to binary variables as it is categorical and there is no relationship between integer values.

In [28]:

```
alt.Chart(source).mark_bar().encode(
    x = "device_type:O",
    y = 'count()',
).properties(
    title='Figure 3. which devices were used')
```

Out[28]:

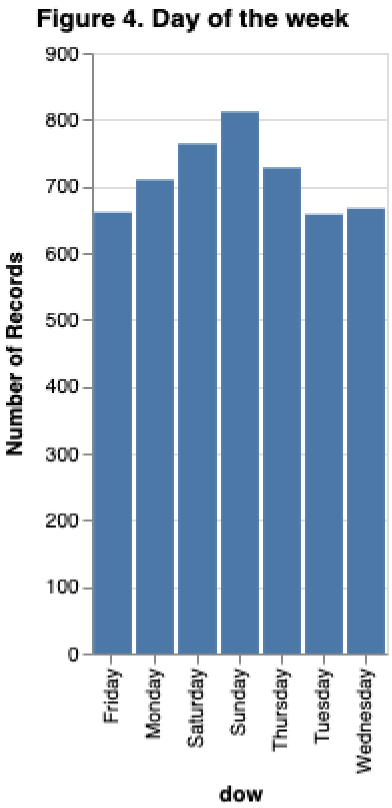


The most common device type is 2. This variable will also need to be converted to binarry variables as it is categorical and there is no relationship between integer values.

In [29]:

```
dataOriginalSource = dataOriginal.sample(n = 5000)
alt.Chart(dataOriginalSource).mark_bar().encode(
    x = "dow:O",
    y = 'count()',
).properties(
    title='Figure 4. Day of the week')
```

Out[29]:



There is a fairly even distribution of days of the week as seen in figure 4.

In [30]:

```
alt.Chart(source).mark_bar().encode(
    x = "day:O",
    y = 'count()',
).properties(
    title='Figure 5. Day recorded in test data')
```

Out[30]:

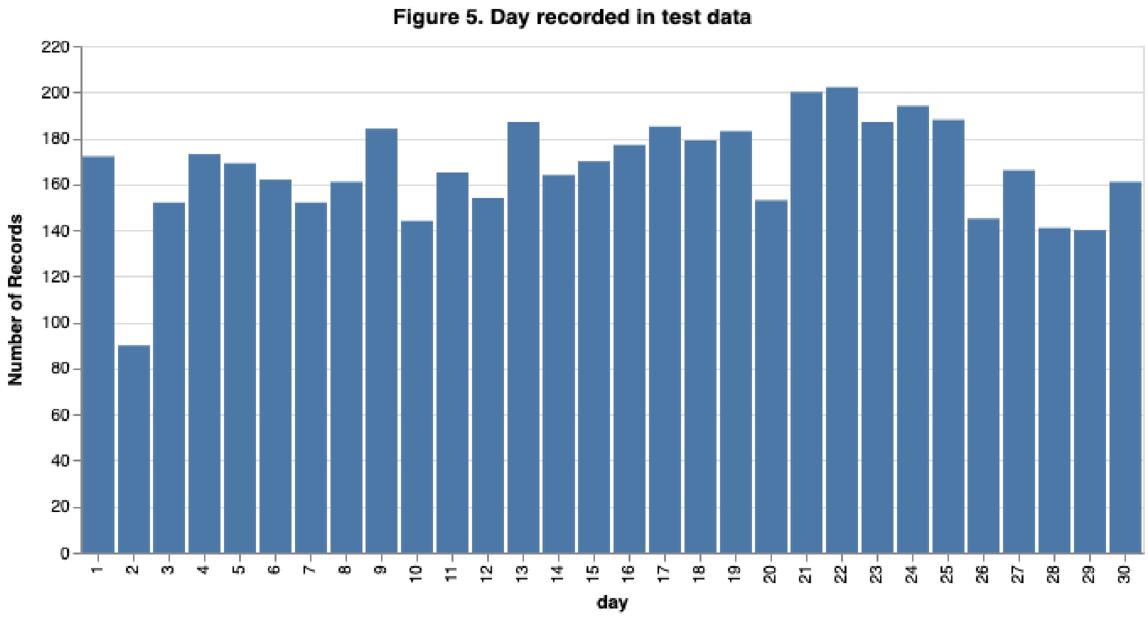


Figure 5 shows the distribution of records to the number of day. This variable will also need to be converted to binarry variables as it is categorical and there is no relationship between integer values.

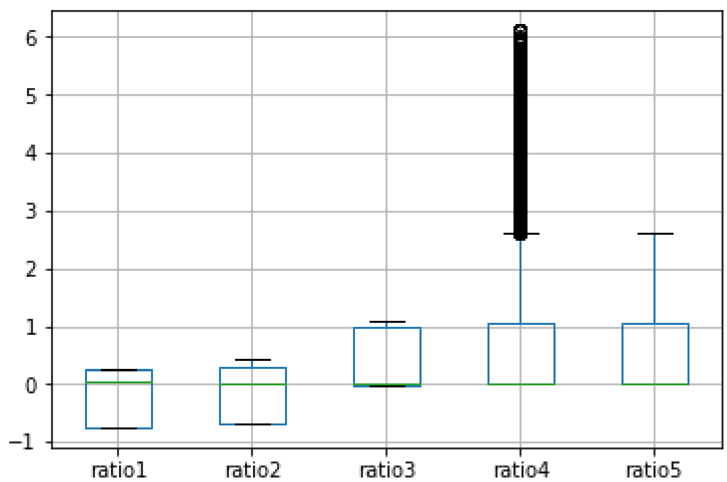
In [117]:

```
print("Figure 6. Distribution of ratio")
source.boxplot(column = ['ratio1','ratio2', 'ratio3','ratio4','ratio5'])
# ratio4 shows significant outliers
```

```
Figure 6
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98b0c32fd0>
```



Outliers can still be observed in ratio 4 in figure 6. These will need to be properly handled before phase II.

In [133]:

```
print("Figure 7. Distribution of Price")
source.boxplot(column = ['price1','price2', 'price3'])
# shows significant outliers for all price points
```

Figure 7. Distribution of Price Variables

Out[133]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f98b18a84e0>

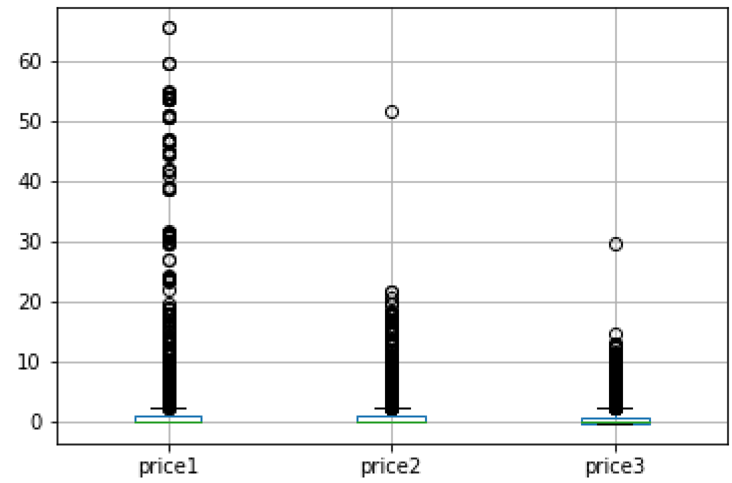


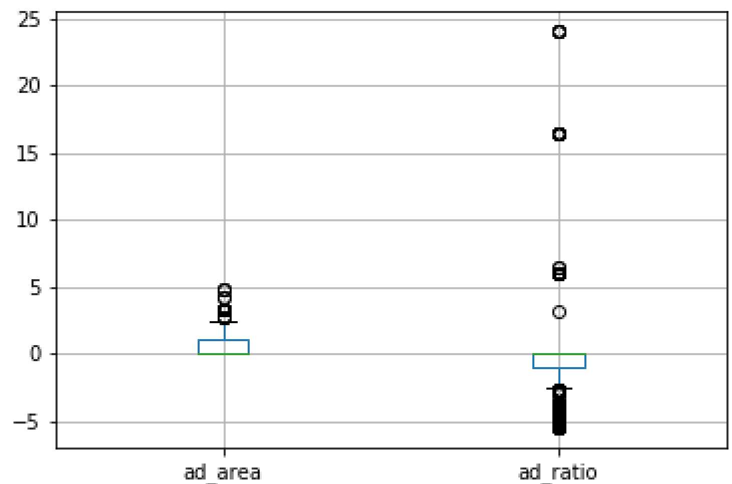
Figure 7 shows the presence of outliers in each of the price variables. These will need to be properly handled before phase II.

In [62]:

```
print("Figure 8. Ad ratio box plot")
source.boxplot(column = ['ad_area','ad_ratio'])
```

Out[62]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f98be73ff28>



There are a small proportion of outliers in ad_area. Ad_ratio also has a lot of outliers. These will need to be properly handled before phase II.

In [71]:

```
print("Figure 9. Distribution of requests")
source.boxplot(column = ['requests'])
```

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98bc3899b0>
```

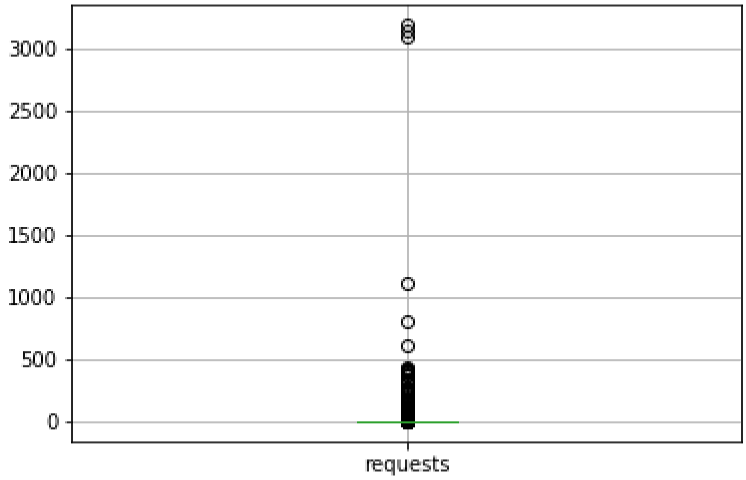


Figure 9 shows that the requests variable has outliers. These will need to be properly handled before phase II.

In [130]:

```
print("Figure 10. CPC, CTR, and viewability boxplot")
source.boxplot(column = ['cpc', 'ctr', 'viewability'])
# shows significant outliers in dataset
```

```
10. CPC, CTR, and viewability boxplot
```

Out[130]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98b185d8d0>
```

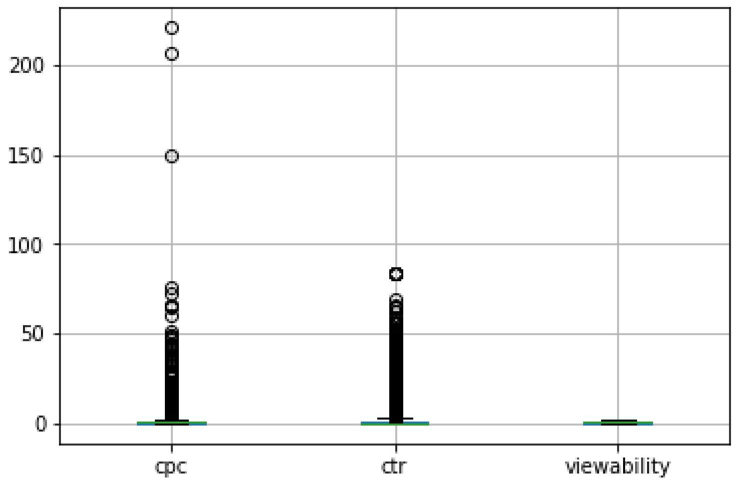


Figure 10 shows that there is one variable without outliers, which is viewability.

In [131]:

```
print("Figure 11. Outliers in impressions")
source.boxplot(column = ['impression'])
```

```
11. Outliers in impressions
```

Out[131]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98b0de5630>
```

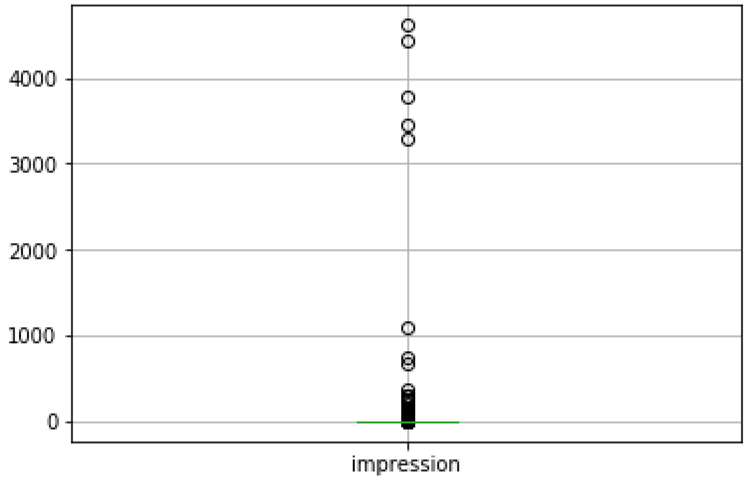


Figure 11 shows that impression has outliers. These will need to be properly handled before phase II.

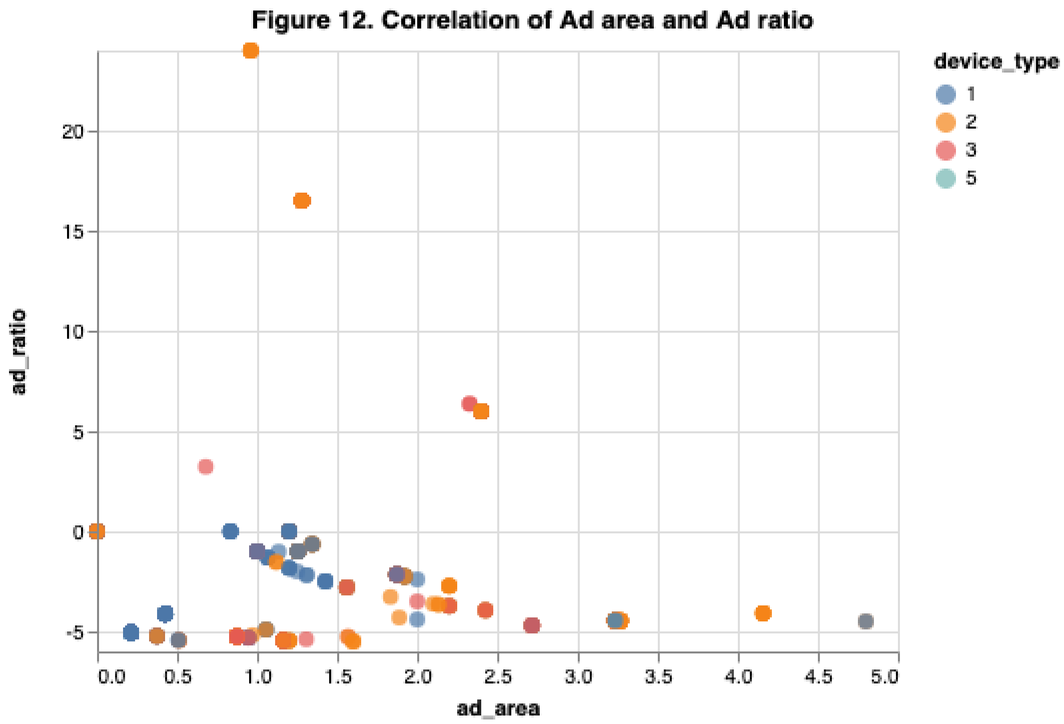
Data Visualisation

Multivariate

In [31]:

```
alt.Chart(source).mark_circle(size=60).encode(
    x='ad_area',
    y='ad_ratio',
    color='device_type:N').properties(
    title='Figure 12. Correlation of Ad area and Ad ratio')
# Ad area and ratio are not independent
```

Out[31]:



In [32]:

```
alt.Chart(source).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='company_id:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['price1', 'price2', 'price3'],
    column=['price3', 'price2', 'price1']
).properties(
    title='Figure 13. Correlation of price')
```

Out[32]:

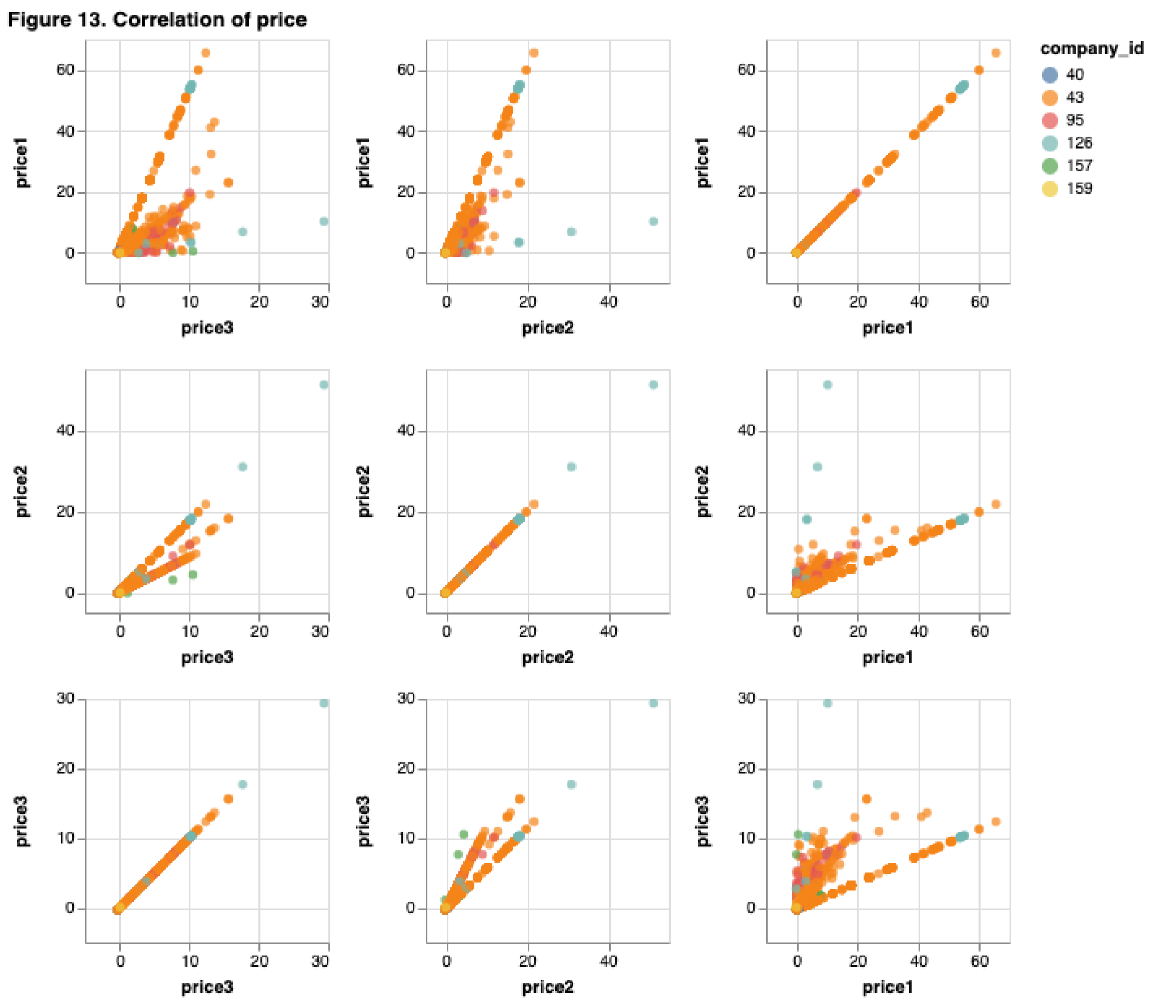
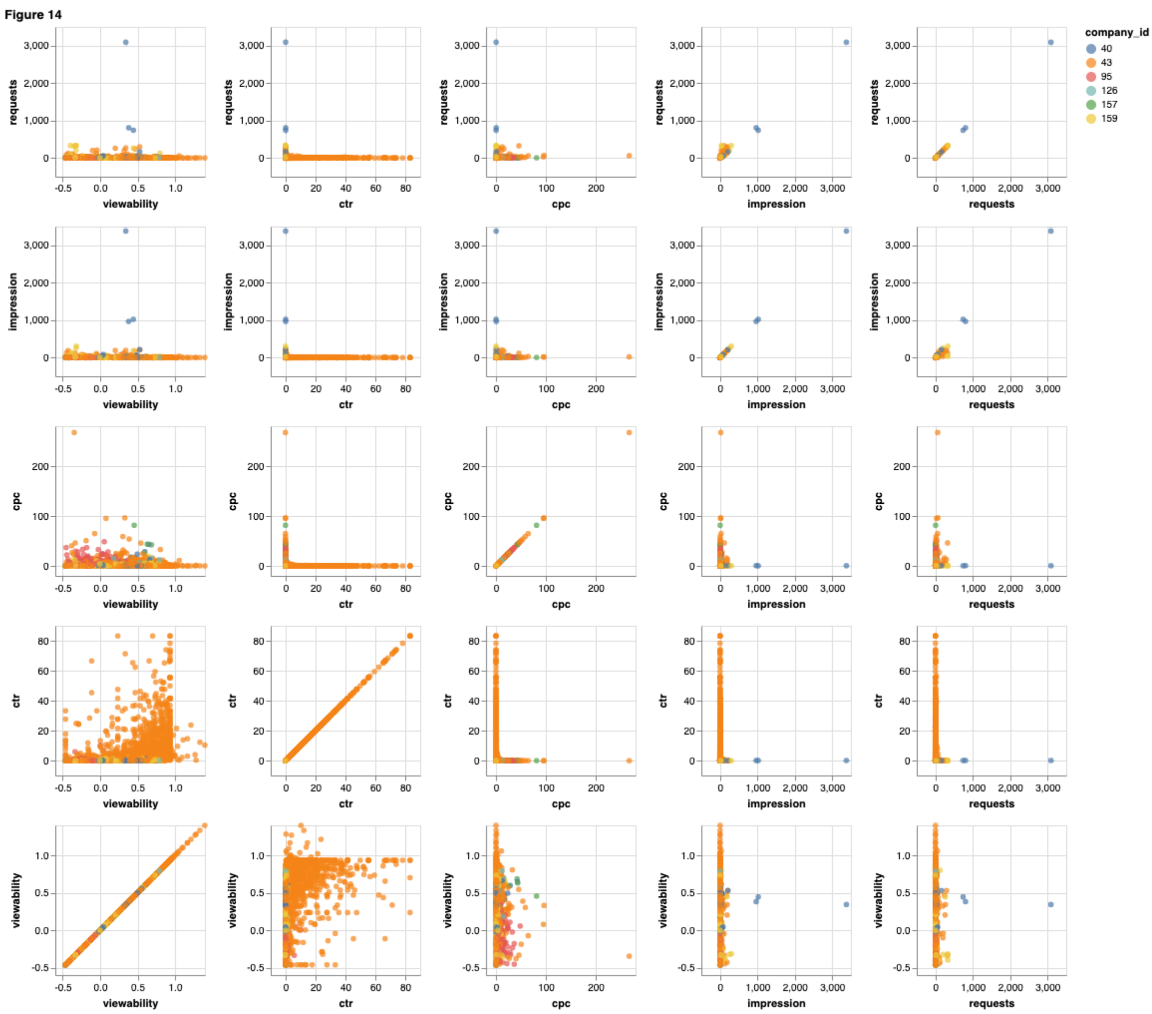


Figure 13: The price variables appear to have a strong positive correlation with each other.

In [33]:

```
alt.Chart(source).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='company_id:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['requests', 'impression', 'cpc', 'ctr', 'viewability'],
    column=['viewability', 'ctr', 'cpc', 'impression', 'requests']
).properties(
    title='Figure 14')
```

Out[33]:

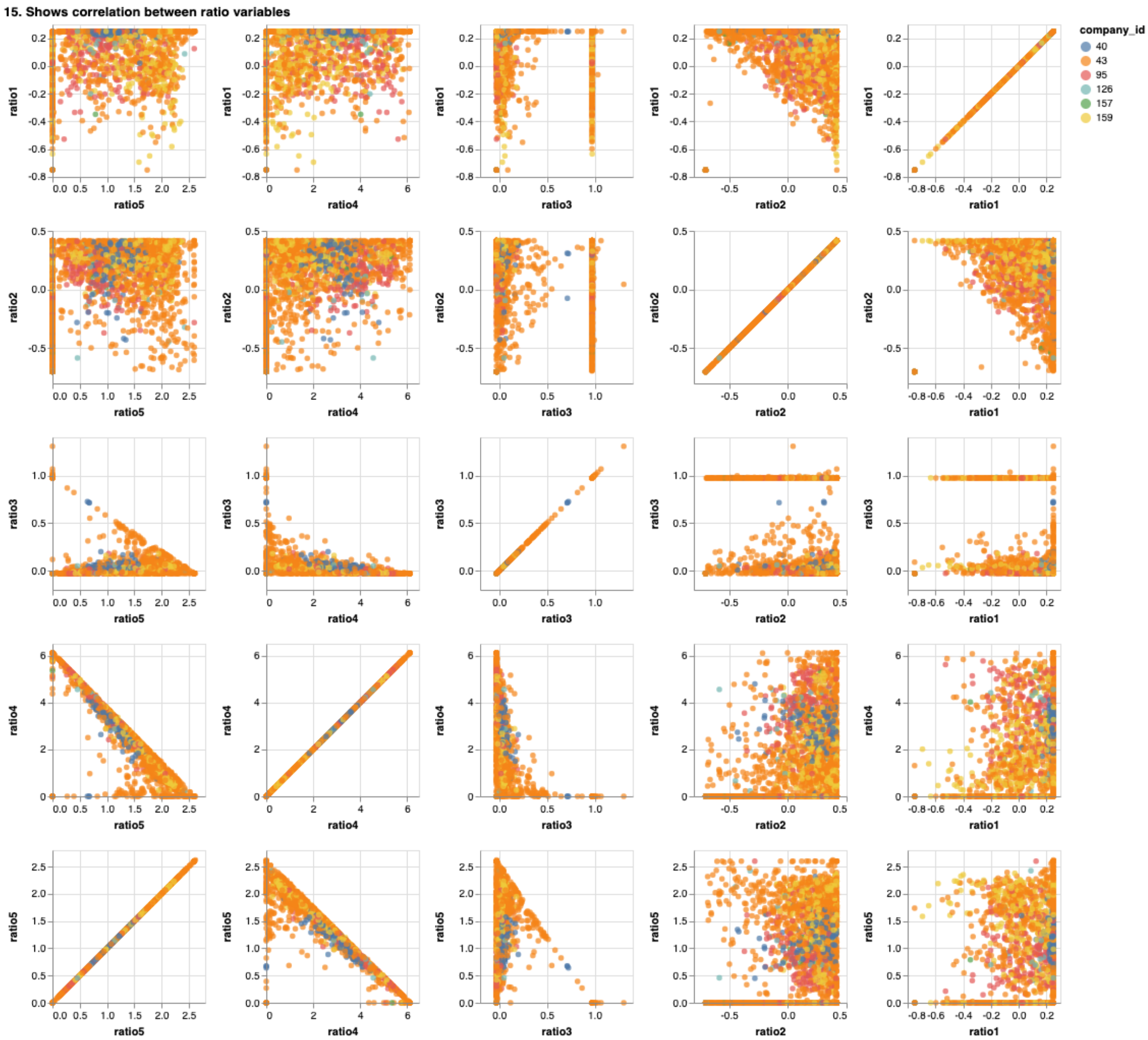


This scatter matrix is somewhat hard to interpret due to the presence of all the outliers which we haven't dealt with properly yet. It appears that some variables are independent, such as cpc and ctr.

In [34]:

```
alt.Chart(source).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='company_id:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['ratio1', 'ratio2', 'ratio3', 'ratio4', 'ratio5'],
    column=['ratio5', 'ratio4', 'ratio3', 'ratio2', 'ratio1']
).properties(
    title='15. Shows correlation between ratio variables')
```

Out[34]:



Conclusion

Further investigation into the data could show the agencies [bidding strategy (https://support.google.com/google-ads/answer/1704424?hl=en)](https://support.google.com/google-ads/answer/1704424?hl=en). Ie. a daily budget, maximum CPC, daily average, enhanced CPC (to maintain a position on Google) or CPA.

Outliers in the data need to be effectively handled before continuing onto the second phase.