BOT DETECTION MODEL

Problem Statement : Build A Model that can detect the Non Human Traffic present in a website!!!

Why do we need to detect Bots ?

No matter how big your website is, you’re almost guaranteed to receive bot traffic at some point. These bots are often up to a range of different things on your site, from indexing web pages to scraping your content. With so many different bots out there, how can you detect bot traffic on your website? And should you be concerned?

Here are 5 reasons that why do need BOT Detection ?

1)Bots can steal your content. You know that content you worked so hard to develop? Your carefully crafted blog posts and pages? All that effort could be wasted in a second if you let bots access your site. Bots can scrape your website for data, information and even pricing in just a matter of minutes. Then, it can be used on other sites, redistributed or even sold for profit in other areas of the web.

2)Bots can slow down your site. Bots bog down your site and overwhelm it with inauthentic, fraudulent traffic. This results in slower page load times for your actual paying customers, which could affect their level of satisfaction or even deter them from buying or visiting altogether.

3)Bots can threaten your website. Malicious bots can hack your website, insert inappropriate links and content, or even crash your site altogether. This can hurt your traffic, your customers and your sales.

4)Bot can take up extra time and money. Many bots spend their time posting spam comments to websites and blogs. While this may not seem like a huge issue, it can be quite frustrating. You’ll have to spend hours each month sorting through these comments to separate the human commenters from the fraudulent ones, which takes you and your resources away from actually running your business. If you don’t remove these spam comments, they end up annoying your readers and possibly leading them away from your site.

5)Bots can mess up your analytics. Analytics are hugely important to a website owner. They tell you how your site is performing, where traffic is coming from and what you might want to tweak throughout the site. Unfortunately, if you have a significant amount of bots accessing your site, this can throw your analytics into upheaval. You won’t have a clear picture of your site’s performance or your next steps for improvement, and you won’t be able to tell what’s real and what’s fake.

The 5-Step Methodology for Spotting Malicious Bot

Activity

1)Separate bots from people

2)Distinguish between browsers and other clients

3)Distinguish between bots within browsers

4)Analyze the payload

5)Determine a target's risk

Import Essential Data

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandas_profiling import ProfileReport
```

In [2]:

```
df = pd.read_csv('ibm_data.csv')
df.head(3)
df.sort_values('VISIT',ascending=False,axis=0)[:5]
```

Out[2]:

	Unnamed: 0	ctry_name	intgrtd_mngmt_name	intgrtd_operating_team_name	
879695	879695	India	India-South Asia	Asia Pacific	BENGA
50894	50894	India	India-South Asia	Asia Pacific	VISAKHAPA
1040366	1040366	United States	United States	North America	LAN
987058	987058	India	India-South Asia	Asia Pacific	KORAMAN
488270	488270	Israel	SPGI	Europe	PETAH

profile = ProfileReport(df, title='Pandas Profiling Report') profile

In [3]:

```
for i in df.columns:
    print('Name of Variable     : ',i)
    print('No. of unique values: ',len(df[i].unique()))
    print(df[i].unique())
    print("\n")
```

```
 'Monaco' 'Grenada' 'Fiji' 'Andorra' 'Zambia' 'Gambia' 'Somalia'
 'Maldives' 'Congo, The Democratic Republic of the' 'Papua New Guinea'
 'Burkina Faso' 'Martinique' 'Cayman Islands' 'Guam' 'Jersey' 'Greenland'
 'New Caledonia' 'Swaziland' 'Congo' 'Guinea' 'Tajikistan'
 'Antigua and Barbuda' 'Burundi' 'Togo' 'Tonga' 'Vanuatu'
 'Virgin Islands, British' 'Saint Kitts and Nevis' 'Guernsey'
 'Saint Vincent and the Grenadines' 'Bhutan' 'Timor-Leste' 'Niger'
 'Iran, Islamic Republic of' 'Sint Maarten (Dutch part)'
 'Turks and Caicos Islands' 'Aruba' 'Gabon' 'Saint Lucia' 'Seychelles'
 'Equatorial Guinea' 'Dominica' 'Lesotho' 'Solomon Islands'
 'Faroe Islands' 'United States Minor Outlying Islands' 'Antarctica'
 'French Guiana' 'Virgin Islands, U.S.' 'Anguilla' 'San Marino'
 'British Indian Ocean Territory' 'Central African Republic' 'Samoa'
 'Holy See (Vatican City State)' 'American Samoa' 'Mayotte' 'Comoros'
 'Cook Islands' 'Saint Martin (French Part)' 'Sao Tome and Principe'
 'Guinea-Bissau']



Name of Variable     :  intgrtd_mngmt_name
No. of unique values:  21
```

#pip install device_detector

In [4]:

```
#Droping similar values
df = df.drop(['wk','mth','yr','Unnamed: 0'],axis=1)
```

In [5]:

```
df["sec_lvl_domn"].value_counts()[:10]
```

Out[5]:

```
COMCAST.NET            26815
RR.COM                 16337
VERIZON.NET            10800
SBCGLOBAL.NET          10783
MYVZW.COM               8285
ACTCORP.IN              6861
COX.NET                 6591
GOOGLE.COM              6269
COMCASTBUSINESS.NET     6041
OCN.NE.JP               5886
Name: sec_lvl_domn, dtype: int64
```

In [6]:

```python
print(df.shape)
df.isnull().sum()
```

```
(1048573, 15)
```

Out[6]:

```
ctry_name                          0
intgrtd_mngmt_name                 0
intgrtd_operating_team_name        0
city                           46586
st                             45445
sec_lvl_domn                  319457
device_type                   842041
operating_sys                   1820
ip_addr                            0
user_agent                         6
VISIT                              0
ENGD_VISIT                         0
VIEWS                              0
page_url                           1
page_vw_ts                      2743
dtype: int64
```

Data Cleaning

In [7]:

```python
#df = df.drop('Unnamed: 0',axis=1)
# Null value is high
df['device_type'] = df['device_type'].fillna('Unknown')
df['city'] = df['city'].fillna('Unknown')
df['st'] = df['st'].fillna('Unknown')
df['sec_lvl_domn'] = df['sec_lvl_domn'].fillna('Unknown')
# Null Value is low
df['operating_sys'] = df['operating_sys'].fillna(df['operating_sys'].mode()[0])
df['page_vw_ts'] = df['page_vw_ts'].fillna(df['page_vw_ts'].mode()[0])
df['page_url'] = df['page_url'].fillna(df['page_url'].mode()[0])
df['user_agent'] = df['user_agent'].fillna(df['user_agent'].mode()[0])
```

In [8]:

```
df.isnull().sum()
#df.describe()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048573 entries, 0 to 1048572
Data columns (total 15 columns):
ctry_name                     1048573 non-null object
intgrtd_mngmt_name            1048573 non-null object
intgrtd_operating_team_name   1048573 non-null object
city                          1048573 non-null object
st                            1048573 non-null object
sec_lvl_domn                  1048573 non-null object
device_type                   1048573 non-null object
operating_sys                 1048573 non-null object
ip_addr                       1048573 non-null object
user_agent                    1048573 non-null object
VISIT                         1048573 non-null int64
ENGD_VISIT                    1048573 non-null int64
VIEWS                         1048573 non-null int64
page_url                      1048573 non-null object
page_vw_ts                    1048573 non-null object
dtypes: int64(3), object(12)
memory usage: 120.0+ MB
```

```
from datetime import datetime, time
#2019-06-04 05:05:18.023100
d = datetime.strptime(df['page_vw_ts'][0], '%Y-%m-%d %H:%M:%S.%f')
t = datetime.strptime(df['page_vw_ts'][1], '%Y-%m-%d %H:%M:%S.%f')

print(d)
print(t)
print(d.timestamp())
print(t.timestamp())
t.timestamp()-d.timestamp()
```

In [9]:

```
df.columns
```

Out[9]:

```
Index(['ctry_name', 'intgrtd_mngmt_name', 'intgrtd_operating_team_name',
       'city', 'st', 'sec_lvl_domn', 'device_type', 'operating_sys', 'ip_add
r',
       'user_agent', 'VISIT', 'ENGD_VISIT', 'VIEWS', 'page_url', 'page_vw_t
s'],
      dtype='object')
```

In [10]:

```
from datetime import datetime, time
#datetime.strptime('2019-06-04 05:05:18.023100', '%Y-%m-%d %H:%M:%S.%f').day
#datetime.strptime('2019-06-04 05:05:18.023100', '%Y-%m-%d %H:%M:%S.%f').month
#datetime.strptime('2019-06-04 05:05:18.023100', '%Y-%m-%d %H:%M:%S.%f').year
df['pg_vw_day']   = df['page_vw_ts'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d %H
df['pg_vw_month'] = df['page_vw_ts'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d %H
df['pg_vw_year']  = df['page_vw_ts'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d %H
df['timeinsecond']= df['page_vw_ts'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d %H
df_1 = df.drop('page_vw_ts',axis = 1)
df_1.head(3)
```

Out[10]:

	ctry_name	intgrtd_mngmt_name	intgrtd_operating_team_name	city	st	
0	United States	United States	North America	SLIDELL	LOUISIANA	
1	Japan	Japan	Japan	TOKYO	TOKYO	
2	United States	United States	North America	ELK GROVE	CALIFORNIA	COM(

In [11]:

```
df['pg_vw_month'] = df['pg_vw_month'].astype(str)
df['pg_vw_year'] = df['pg_vw_year'].astype(str)
df['period'] = df['pg_vw_month'].str.cat(df['pg_vw_year'], sep =".")
df['pg_vw_month'] = df['pg_vw_month'].astype(int)
df['pg_vw_year'] = df['pg_vw_year'].astype(int)
#df['period']
```

In [12]:

```
max_t = df_1['timeinsecond'].max()
min_t = df_1['timeinsecond'].min()
print(round(max_t/(60*60*24)))
print(round(min_t/(60*60*24)))
d = max_t-min_t
print(d)
print(datetime.fromtimestamp(min_t))
print(datetime.fromtimestamp(max_t))
```

```
18415.0
11687.0
581283199.0512999
2001-12-31 16:33:14.028500
2020-06-02 12:06:33.079800
```

Bot Detection

First, we need to detect the bot

In [13]:

```
df.head(3)
```

Out[13]:

	ctry_name	intgrtd_mngmt_name	intgrtd_operating_team_name	city	st	
0	United States	United States	North America	SLIDELL	LOUISIANA	
1	Japan	Japan	Japan	TOKYO	TOKYO	
2	United States	United States	North America	ELK GROVE	CALIFORNIA	COM(

In [14]:

```python
mask = (((3*df['VISIT'])<df['ENGD_VISIT']) | ((3*df['VISIT'])<df['VIEWS']))
print(mask.unique())
mask.value_counts()#describe()
```

```
[False  True]
```

Out[14]:

```
False    1028368
True       20205
dtype: int64
```

In [15]:

```python
df['bot'] = np.where(mask,1,0)
df['bot'].value_counts()
```

Out[15]:

```
0    1028368
1      20205
Name: bot, dtype: int64
```

In [16]:

```python
df_b1 = df.copy()
df_b1 = df_b1[['ip_addr','VISIT','ENGD_VISIT','VIEWS','bot']]
print(df_b1.shape)
#df_b1.head(2)
df_b1.loc[df_b1['bot']==1].head()
```

```
(1048573, 5)
```

Out[16]:

	ip_addr	VISIT	ENGD_VISIT	VIEWS	bot
40	9411effb3d1c9644d4bc14f092d69428aba2d8cb834594...	1	0	6	1
148	7a5107056137f31d11f88e15107bf1595b81b6802f33ff...	1	1	4	1
207	867c245be29981ab7e05a45f3d55822d05f2345695d0e2...	1	1	4	1
252	5f57ff161243f89daa1dad4ea6821edeb004f31d8e9954...	1	1	6	1
258	f4b7f51cfa55482c87fe09393be43defada3ed136a6b9a...	2	2	15	1

In [17]:

```python
ip_high_fre = df_b1['ip_addr'].value_counts()
ip_high_fre[:5]
```

Out[17]:

```
b8a8233899cbd26ba1e2207af00ed76f6d3a2393f9967aa43bd4676d264a2894    1777
0ec667e87013e1398ddaa61c14f7118f0dd6adc9abc2a2a0608545b59960fdc9     735
1efe766ebdcd5b9065d8ae9c5e2070201d8ca83768809009f4da5a78dc8acc14     598
c97c81d31fb39cbba79f371dcc09accdb345084c0eb0a3efc146446bce14aead     511
a652f03df170d22eaeb7c665ddd4f7caf5b715adfcca4bd719188e0c3479af5f     476
Name: ip_addr, dtype: int64
```

In [18]:

```
df_gb = df_b1.groupby(['ip_addr'],axis=0).agg({'VISIT':np.sum,'ENGD_VISIT':np.sum,'VIE
df_gb['bot'] = df_gb['bot'].replace(range(1,df_gb['bot'].max()+1),1)
print(df_gb['bot'].unique())
print(df_gb.shape)
df_gb.sort_values('bot',ascending=False)[:5]
```

```
[0 1]
(450602, 5)
```

Out[18]:

	ip_addr	VISIT	ENGD_VISIT	VIEWS	bot
439980	f9ea77612c10d736c2e3744a6623bcd11c5a6c9be1fd2e...	1	1	4	1
16423	095dce82441b60e612abd10ef2f356451b9387127eeed2...	1	0	4	1
237246	86f3fdd80cd80cf2bdb67279ea07095123bf45dd9e6759...	62	27	75	1
312462	b1ad289e7ece0453d6778c1042ff69a8c7be723c00323f...	26	11	39	1
422017	efb91c596e577ab5081b5428fa20dc8302f4bc04eb0799...	20	11	30	1

In [19]:

```
df_gb1 = df_gb.copy()
mask = (((3*df_gb1['VISIT'])<df_gb1['ENGD_VISIT']) | ((3*df_gb1['VISIT'])<df_gb1['VIEW
print(mask.unique())
mask.value_counts()
```

```
[False  True]
```

Out[19]:

```
False    447471
True       3131
dtype: int64
```

In [20]:

```
print(df_gb['bot'].value_counts())
df_gb1['bot'] = np.where(mask,1,0)
df_gb1['bot'].value_counts()
```

```
0    436771
1     13831
Name: bot, dtype: int64
```

Out[20]:

```
0    447471
1      3131
Name: bot, dtype: int64
```

For bot detection, our peiority is to detect all bot even if there is wrong

In [21]:

```
mg = pd.merge(df_gb,df_gb1,left_on=df_gb['ip_addr'],right_on=df_gb1['ip_addr'])
mg[['ip_addr_x','bot_x','bot_y']].head()
```

Out[21]:

	ip_addr_x	bot_x	bot_y
0	00000bfd838fedd4c2adff293a64d5efa3406b27053a10...	0	0
1	00002f7fddaa273634e38ec7004224498baa7a46cf2640...	0	0
2	0000694257f2882ccebb6a5431a8ebca19063252492891...	0	0
3	0000a229c86f8baa193ffdcf9c2a88f83a91b9faad73f7...	0	0
4	0000faf7927142fa7fbc189d4a1bf23e4eb0c8d3b49c62...	0	0

In [22]:

```
mg['bot_1'] = mg['bot_y']+mg['bot_y']
mg['bot_1'] = mg['bot_1'].replace(range(1,mg['bot_1'].max()+1),1)
mg['bot_1'].sort_values(ascending=False)[:5]
df_gb2 = mg[['ip_addr_x','bot_1']]
df_gb2.head(3)
```

Out[22]:

	ip_addr_x	bot_1
0	00000bfd838fedd4c2adff293a64d5efa3406b27053a10...	0
1	00002f7fddaa273634e38ec7004224498baa7a46cf2640...	0
2	0000694257f2882ccebb6a5431a8ebca19063252492891...	0

In [23]:

```
df.head(1)
```

Out[23]:

	ctry_name	intgrtd_mngmt_name	intgrtd_operating_team_name	city	st	sec_l
0	United States	United States	North America	SLIDELL	LOUISIANA	CHART

1 rows × 21 columns

In [24]:

```
df_1 = pd.merge(df,df_gb2,how='left',right_on='ip_addr_x',left_on='ip_addr',copy=False
#df_1.head(3)
```

In [25]:

```
df_1['Bot'] = df_1['bot']+df_1['bot_1']
df_1 = df_1.drop(['bot','bot_1','ip_addr_x'],axis=1)
df_1['Bot'] = df_1['Bot'].replace(range(1,df_1['Bot'].max()+1),1)
df_1.sort_values('Bot',ascending=False)[:3]
#df_1.head(3)
```

Out[25]:

nt	...	ENGD_VISIT	VIEWS	page_url	page_vw_ts	pg_
.0 1) ...	...	3	7	www-06.ibm.com/ibm/support/jp/dhrm/dhrmbp.nsf/...	2019-06-04 15:21:27.077200	
.0 0; ...	...	1	5	www.ibm.com/support/knowledgecenter/ss6pew_9.5...	2019-06-04 06:22:13.074500	
E; _1 ...	...	0	4	www.ibm.com/thought-leadership/smart	2019-06-04 14:16:42.046600	

Sample data ¶

Data is so large & it takes long time to compute. So, im taking sample from that data

In [31]:

```
sam_data = df_1.sample(frac=.25,random_state=1)
print(sam_data.shape)
sam_data['Bot'].value_counts()
```

```
(10486, 21)
```

Out[31]:

```
0    10205
1      281
Name: Bot, dtype: int64
```

High Frequency of visit based on month

In [32]:

```
df_p = sam_data[['ip_addr','period','VISIT']]
df_pp = pd.pivot_table(df_p,values='VISIT',index =['ip_addr'],columns =['period'], agg
df_pp["sum_visit"] = df_pp.sum(axis = 1, skipna = True)
df_pp[:2]
```

Out[32]:

period	ip_addr	11.2018	12.2018	6.2019	sum_visit
0	0003855a3aba0f8b3059208ae801e09a67f2b155a6c63a...	NaN	NaN	1.0	1.0
1	0006e8a3008f5ebb5f00341661cbdd6d92310f3691ad0e...	NaN	NaN	1.0	1.0

In [33]:

```
df_pp['bot'] = 0
for i in df_p['period']:
    mask = ((df_pp[i]>10) | (df_pp['sum_visit']>40))
    df_pp['bot'] = np.where(mask,1,0)

df_c = df_pp[['ip_addr','bot']]
df_pp[:1]
```

Out[33]:

period	ip_addr	11.2018	12.2018	6.2019	sum_visit
0	0003855a3aba0f8b3059208ae801e09a67f2b155a6c63a...	NaN	NaN	1.0	1.0

In [34]:

```
df_2 = pd.merge(sam_data,df_c,how='left',right_on='ip_addr',left_on='ip_addr',copy=Fal
df_2.head(2)
```

Out[34]:

	ctry_name	intgrtd_mngmt_name	intgrtd_operating_team_name	city	
0	Italy	Italy	Europe	ROMA	RO
1	United Kingdom	UKI	Europe	CLYDEBANK	WE DUNBARTONSHI

2 rows × 22 columns

In [35]:

```
df_2['Bot_data'] = df_2['Bot']+df_2['bot']
df_2['Bot_data'] = df_2['Bot'].fillna(0)
df_2['Bot_data'] = df_2['Bot'].astype(int)
df_2 = df_2.drop(['Bot','bot'],axis=1)
```

In [36]:

```
df_2['Bot_data'] = df_2['Bot_data'].replace(range(1,df_2['Bot_data'].max()+1),1)
df_2.sort_values('Bot_data',ascending=False)[:3]
df_2.shape
df_2['Bot_data'].value_counts()
```

Out[36]:

```
0    10205
1      281
Name: Bot_data, dtype: int64
```

In [37]:

```
sam_data['Bot'].value_counts()
```

Out[37]:

```
0    10205
1      281
Name: Bot, dtype: int64
```

In [38]:

```
data = df_2.copy()
data[:2]
```

Out[38]:

	ctry_name	intgrtd_mngmt_name	intgrtd_operating_team_name	city	
0	Italy	Italy	Europe	ROMA	RO
1	United Kingdom	UKI	Europe	CLYDEBANK	WE DUNBARTONSHI

2 rows × 21 columns

Ip_addr which has high frequency is identified as bot

Hence, bot were detected

Visualization

plt.figure(figsize=[20,20]) sns.scatterplot(y = 'ip_addr',x = 'period',hue='Bot',data=sam_data)

#pip install user-agents

In [39]:

```
for i in data.columns:
    print('Name of Variable    : ',i)
    print('No. of unique values: ',len(data[i].unique()))
    print(data[i].unique())
    print("\n")
```

```
 'Liechtenstein' 'Togo' 'Armenia' 'Somalia' 'Iraq' 'Mauritius' 'Nicaragua'
 'Senegal' 'Myanmar' 'Malta' 'Rwanda' 'Uzbekistan' 'Fiji' 'Bahamas'
 'Haiti' "C�te d'Ivoire" 'Libya' 'Lebanon' 'Seychelles' 'Sierra Leone'
 'Barbados' 'Maldives' 'Guadeloupe']


Name of Variable    :  intgrtd_mngmt_name
No. of unique values:  21
['Italy' 'UKI' 'DACH' 'France' 'United States' 'SSA'
 'Middle East & Africa' 'ASEAN' 'Canada' 'Mexico' 'CEE' 'Greater China'
 'India-South Asia' 'BeNeLux' 'Australia/NZ' 'Japan' 'Brazil' 'Korea'
 'SPGI' 'Nordic' 'Unassigned']


Name of Variable    :  intgrtd_operating_team_name
No. of unique values:  8
['Europe' 'North America' 'Latin America' 'Middle East & Africa'
 'Asia Pacific' 'Greater China Group' 'Japan' 'Unassigned']
```

1)Separate bots from people

In [40]:

```
bot_data = data.loc[data['Bot_data']==1]
peo_data = data.loc[data['Bot_data']==0]
```

In [41]:

```
bot_data.head(2)
```

Out[41]:

_agent	...	ENGD_VISIT	VIEWS	page_url	page_vw_ts	pg_vw_da
.LA/5.0 DOWS NT 6.1; 4; X64; ₹V:65...	...	0	1	www-01.ibm.com/support/docview.wss? uid=swg2126...	2019-06-04 19:22:29.070800	
.LA/5.0 DOWS T 10.0; 4; X64; RV:6...	...	1	5	developer.ibm.com/recipes/tutorials/setup-priv...	2019-06-04 05:51:00.024500	

In [42]:

```
peo_data.head(1)
```

Out[42]:

user_agent	...	ENGD_VISIT	VIEWS	page_url	page_vw_ts	pg_vw_da
MOZILLA/5.0 (IPHONE; CPU IPHONE OS 12_1_4 LIKE...	...	0	1	www-03.ibm.com/press/it/it/resources.wss	2019-06-04 09:17:12.070600	

2)Distinguish between browsers and other clients

In [58]:

```
from device_detector import DeviceDetector
from device_detector import SoftwareDetector
data_bro = data[['ip_addr','user_agent','Bot_data','sec_lvl_domn','device_type','opera
```

In [59]:

```
#data['bot_1'] = data['user_agent'].apply(lambda x: SoftwareDetector(x).parse().is_bot
data_bro['Browser']    = data_bro['user_agent'].apply(lambda x: DeviceDetector(x).parse
data_bro['Browser'].value_counts()[:2]
```

```
C:\Users\Anand\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
```

Out[59]:

```
Chrome               5741
Internet Explorer    1035
Name: Browser, dtype: int64
```

In [60]:

```
data_bro1 = data_bro.copy()
data_bro_cnt = data_bro['Browser'].value_counts().reset_index()
unknown_bro = data_bro_cnt.loc[data_bro_cnt['Browser']<2]["index"].tolist()
unknown_bro.append('')
print(unknown_bro)
```

```
['Opera Devices', 'Firefox Focus', 'PERFICIENT_PERF', 'Baidu Browser', ';__C
T_JOB_ID__:', 'Iron', 'Pinterest', '']
```

In [61]:

```
data_bro[:2]
```

Out[61]:

	ip_addr	user_agent	Bot_data	sec_lvl_domn	
0	6984db784d5c2b12af23ed7dbccb5511a4038b9da629b0...	MOZILLA/5.0 (IPHONE; CPU IPHONE OS 12_1_4 LIKE...	0	Unknown	M
1	7e34597f49144ed22808eaa80aeb8602bb455df0df5e65...	MOZILLA/5.0 (LINUX; ANDROID 6.0.1; SM-N910F) A...	0	VIRGINM.NET	M

In [62]:

```
data_bro["Bot_data"].value_counts()
```

Out[62]:

```
0    10205
1      281
Name: Bot_data, dtype: int64
```

In [63]:

```
print(unknown_bro)
print(data_bro['Browser'].unique())
#mask = (len(data_bro['Browser'])>20 | (data_bro['Browser']==''))
#data_bro['Bad_bot'] = np.where(mask,1,0)
#data_bro[:5]
```

```
['Opera Devices', 'Firefox Focus', 'PERFICIENT_PERF', 'Baidu Browser', ';__C
T_JOB_ID__:', 'Iron', 'Pinterest', '']
['Mobile Safari' 'Chrome Mobile' 'Firefox' 'Chrome' 'Internet Explorer'
 'Samsung Browser' 'QQ Browser' 'Microsoft Edge' 'Chrome Webview' 'Safari'
 'Yandex Browser' 'Opera' 'Sogou Explorer' 'Facebook' 'MIUI Browser'
 'Chrome Mobile iOS' 'Android Browser' 'Firefox Mobile' '' 'Apple News'
 'Firefox Mobile iOS' 'WeChat' 'Vivaldi' 'Opera Mobile' 'Maxthon'
 'BlackBerry Browser' 'Chromium' 'Mobile Silk' 'Baidu Spark' 'UC Browser'
 'Liebao' 'Pinterest' 'Opera Devices' 'Yahoo! Japan' 'Oppo Browser'
 'Baidu Browser' 'Firefox Focus' 'Facebook Messenger' 'Coc Coc'
 'CM Browser' 'Iron' 'PERFICIENT_PERF' 'NetFront' ';__CT_JOB_ID__:']
```

In [64]:

```
def Filter(string, substr):
    return [str for str in string if
              any(sub in str for sub in substr)]
#doc = list(filter(lambda x: sub_doc in x, e))
```

In [93]:

```
n = []
for i in range(20):
    n.append(i)

n = str(n)
valid_bro = Filter(data_bro['Browser'].unique(),n)
valid_bro[:5]
```

Out[93]:

```
['Mobile Safari',
 'Chrome Mobile',
 'Internet Explorer',
 'Samsung Browser',
 'QQ Browser']
```

In [66]:

```
data_bro['who'] = data_bro['Bot_data']
ma = ((data_bro['Browser']=='Mobile Safari')& (data_bro['Bot_data']==1))
data_bro.loc[ma,'who']=2
data_bro[:3]
# 0 --> People
# 1-->bad
# 2 --> Good
```

```
C:\Users\Anand\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  """Entry point for launching an IPython kernel.
C:\Users\Anand\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  self.obj[item] = s
```

Out[66]:

	ip_addr	user_agent	Bot_data	sec_lvl_domn	
0	6984db784d5c2b12af23ed7dbccb5511a4038b9da629b0...	MOZILLA/5.0 (IPHONE; CPU IPHONE OS 12_1_4 LIKE...	0	Unknown	M
1	7e34597f49144ed22808eaa80aeb8602bb455df0df5e65...	MOZILLA/5.0 (LINUX; ANDROID 6.0.1; SM-N910F) A...	0	VIRGINM.NET	M
2	790f5c5ea63b56f9ebdd33cf8c6b4e4449c1834dad128c...	MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:5...	0	GC-GRUPPE.DE	

In [67]:

```
for i in valid_bro:
    ma = ((data_bro['Browser']==i)& (data_bro['Bot_data']==1))
    data_bro.loc[ma,'who']= 2
```

In [68]:

```
data_bro["who"].value_counts()
```

Out[68]:

```
0    10205
1      203
2       78
Name: who, dtype: int64
```

3)Distinguish between bots within browsers

In [69]:

```
data_browser = data_bro.copy()
data_browser["Who_is_it"] = data_browser["who"]
data_browser["Who_is_it"] = data_browser["Who_is_it"].replace({0:'People',1:'Bad',2:'G
data_browser[:2]
```

Out[69]:

	ip_addr	user_agent	Bot_data	sec_lvl_domn	
0	6984db784d5c2b12af23ed7dbccb5511a4038b9da629b0...	MOZILLA/5.0 (IPHONE; CPU IPHONE OS 12_1_4 LIKE...	0	Unknown	M
1	7e34597f49144ed22808eaa80aeb8602bb455df0df5e65...	MOZILLA/5.0 (LINUX; ANDROID 6.0.1; SM-N910F) A...	0	VIRGINM.NET	M

plt.figure(figsize=[20,20]) sns.scatterplot(y = 'Browser',x = 'Who_is_it',data=data_browser)

plt.figure(figsize=[20,20]) sns.scatterplot(y = 'ip_addr',x = 'Who_is_it',data=data_browser)

Most of the bots where from chrome

4)Analyze the payload

In [70]:

```
data_browser[:3]
```

Out[70]:

ip_addr	user_agent	Bot_data	sec_lvl_domn	device_typ
784d5c2b12af23ed7dbccb5511a4038b9da629b0...	MOZILLA/5.0 (IPHONE; CPU IPHONE OS 12_1_4 LIKE...	0	Unknown	MOBILEPHONE
7f49144ed22808eaa80aeb8602bb455df0df5e65...	MOZILLA/5.0 (LINUX; ANDROID 6.0.1; SM-N910F) A...	0	VIRGINM.NET	MOBILEPHONE
5ea63b56f9ebdd33cf8c6b4e4449c1834dad128c...	MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:5...	0	GC-GRUPPE.DE	Unknown

In [ ]:

```
 
```

In [76]:

```
data_browser.pivot_table(index="period", columns='Who_is_it', aggfunc='size', fill_valu
```

Out[76]:

who	period	0	1	2
0	11.2018	21	3	0
1	12.2018	1	0	0
2	6.2019	10183	200	78

In [82]:

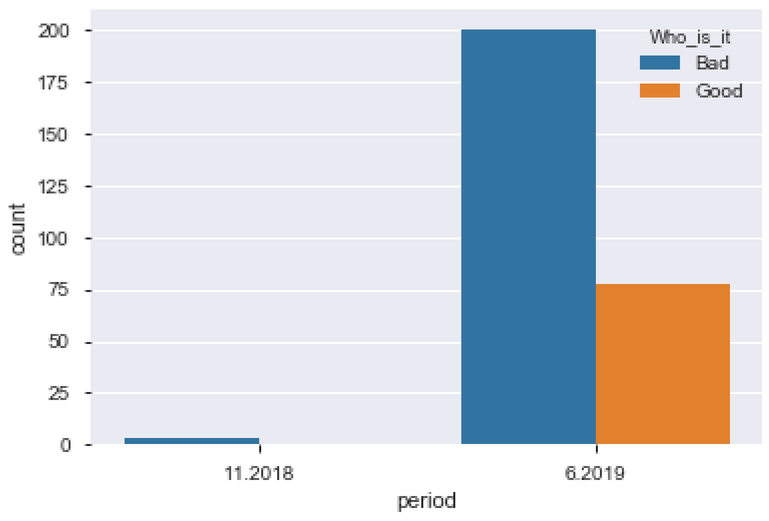
```
dta_bt = data_browser.loc[data_browser['Bot_data']==1]
```

In [83]:

```
sns.countplot(x="period",hue="Who_is_it",data=dta_bt)
```

Out[83]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b09e5d6080>
```



5)Determine a target's risk

In [84]:

```
data_browser[:3]
```

Out[84]:

ip_addr	user_agent	Bot_data	sec_lvl_domn	device_typ
784d5c2b12af23ed7dbccb5511a4038b9da629b0...	MOZILLA/5.0 (IPHONE; CPU IPHONE OS 12_1_4 LIKE...	0	Unknown	MOBILEPHONE
7f49144ed22808eaa80aeb8602bb455df0df5e65...	MOZILLA/5.0 (LINUX; ANDROID 6.0.1; SM-N910F) A...	0	VIRGINM.NET	MOBILEPHONE
5ea63b56f9ebdd33cf8c6b4e4449c1834dad128c...	MOZILLA/5.0 (WINDOWS NT 10.0; WIN64; X64; RV:5...	0	GC-GRUPPE.DE	Unknown

In [91]:

```
sec = data_browser.pivot_table(index="sec_lvl_domn", columns='Who_is_it', aggfunc='siz
sec.sort_values("Bad",ascending=False)[:7]
```

Out[91]:

Who_is_it	sec_lvl_domn	Bad	Good	People
2582	Unknown	55	28	3151
514	COMCAST.NET	4	3	239
2529	UNINET-IDE.COM.MX	3	1	21
2036	SBCGLOBAL.NET	3	1	101
1761	OPTONLINE.NET	3	0	37
1993	RR.COM	3	0	144
2603	VERIZON.NET	3	3	96

The Bad bot can be detected using:

- Frequent high visit with unknown browser & improper user agent