In [118]:
```
## Importing required packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
```

In [102]:
```
## Importing data

data = pd.read_csv("C:\\Users\\Snigdha.Cheekoty\\Downloads\\daily_revenue.csv"
)
```

In [103]:
```
type(data)

### Checking the dattype of the input: Pandas dataframe
```

Out[103]:
```
pandas.core.frame.DataFrame
```

In [104]:
```
data.head(20)
## checking the first 20 records
```

Out[104]:

	date	site	revenue	pageviews
0	7/13/2016	wearewearside	0.539353	389
1	7/13/2016	projectspurs	3.588072	2353
2	7/13/2016	totalbarca	5.130714	1228
3	7/14/2016	projectspurs	4.264064	2762
4	7/14/2016	totalbarca	2.280724	539
5	7/14/2016	hoosierhuddle	0.219857	187
6	7/14/2016	wearewearside	0.895627	638
7	7/15/2016	hoosierhuddle	0.132319	173
8	7/15/2016	wearewearside	0.064656	71
9	7/15/2016	totalbarca	2.157703	785
10	7/15/2016	projectspurs	2.853130	2844
11	7/16/2016	projectspurs	1.683579	2257
12	7/16/2016	wearewearside	0.390841	577
13	7/16/2016	hoosierhuddle	0.068215	120
14	7/16/2016	totalbarca	1.373814	672
15	7/17/2016	projectspurs	0.841798	1180
16	7/17/2016	wearewearside	0.507751	784
17	7/17/2016	hoosierhuddle	0.081240	150
18	7/17/2016	totalbarca	1.296636	663
19	7/18/2016	therepublikofmancunia	3.052134	2524

In [110]:
```
## Importing subtted data for EDA
data2 = pd.read_csv("C:\\Users\\Snigdha.Cheekoty\\OneDrive - Serco\\Desktop\\monthlydata.csv")
```

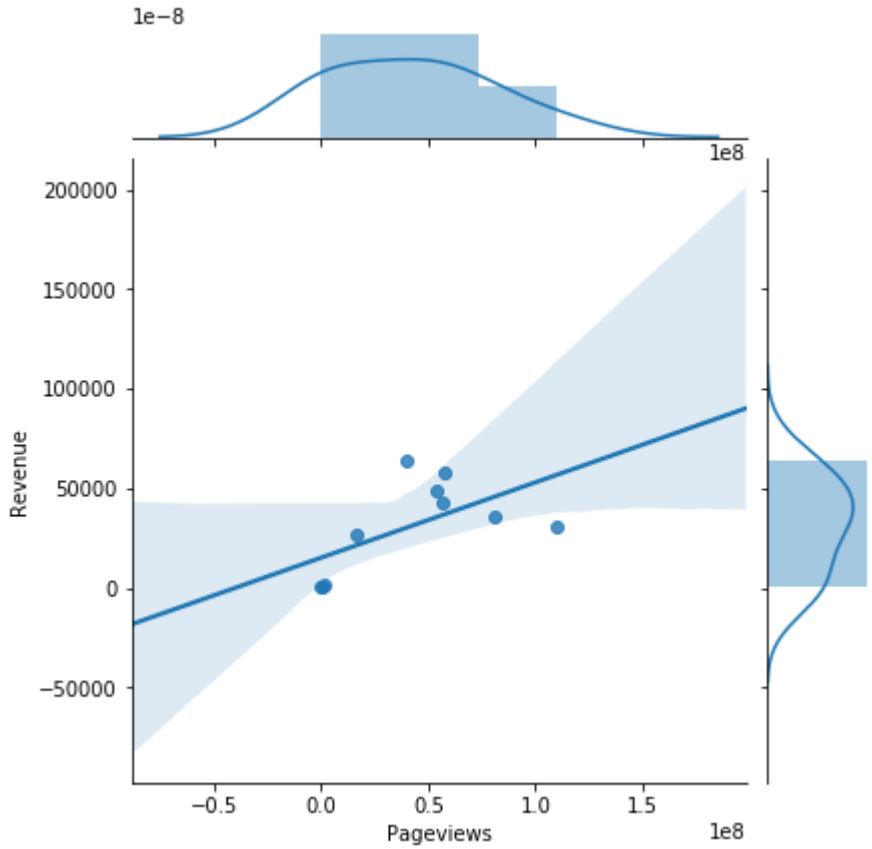
In [114]:
```
data2
## Time-based(Monthly) data for revenue and pageviews
```

Out[114]:

	Month	Pageviews	Revenue
0	July	264657	196
1	August	550	257
2	October	1512223	1735
3	November	16313638	26570
4	December	39452572	63920
5	January	57822021	57601
6	February	57130139	42665
7	March	110093915	30548
8	April	81325976	36148
9	May	53561098	49224

In [125]:
```
sns.jointplot(x = "Pageviews", y = "Revenue", data  = data2 , kind = "reg")
```

Out[125]: <seaborn.axisgrid.JointGrid at 0x1e281d137f0>



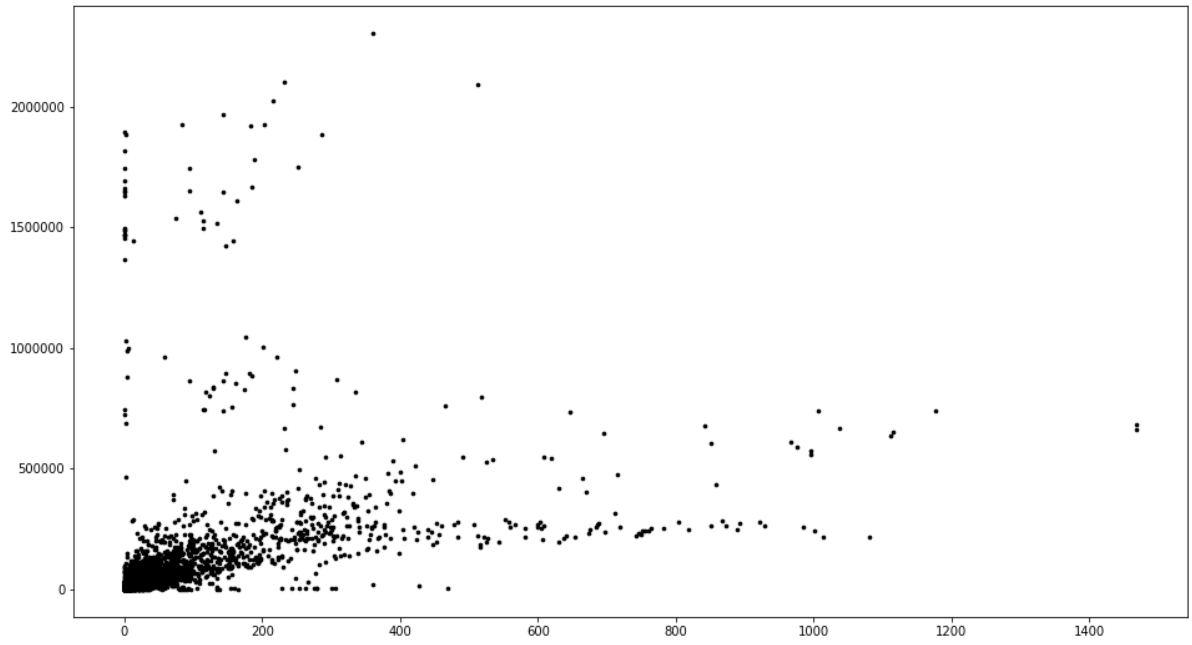
In [ ]:

In [ ]:

In [64]:
```
## importing library for kmeans clustering
from sklearn.cluster import KMeans
```

In [65]:
```
# Obtaining the values and Plotting them
f1 = data['revenue'].values
f2 = data['pageviews'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)
```

Out[65]: <matplotlib.collections.PathCollection at 0x1e2eb8c6128>

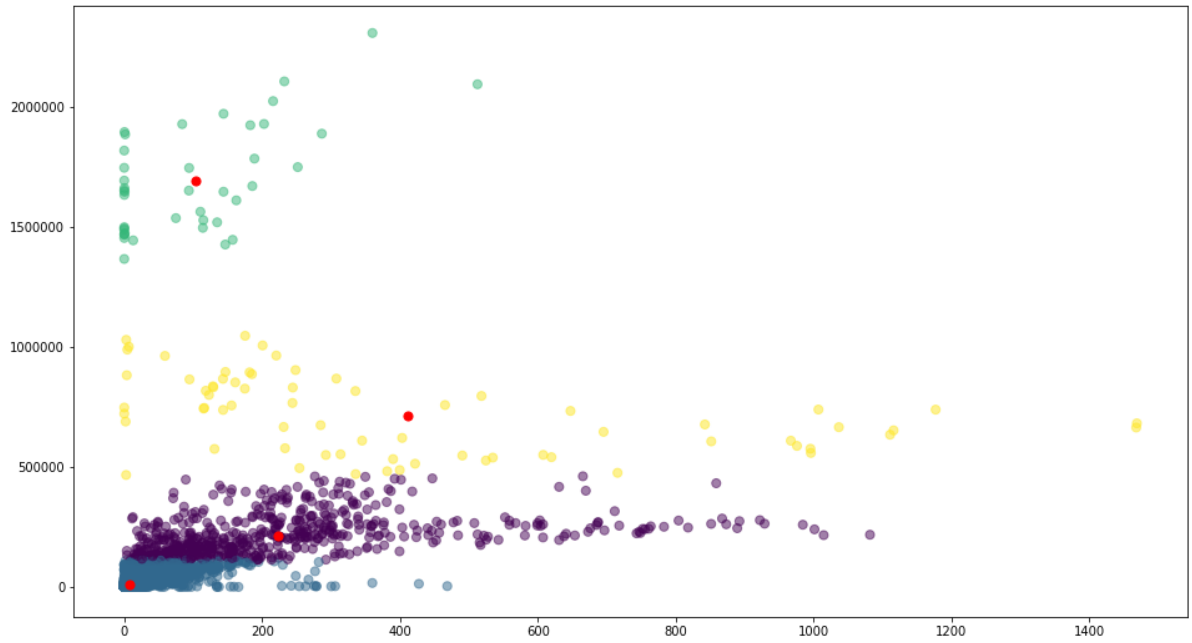


In [66]:
```
# Assigning the Number of clusters
## Choosing the k value after considering an elbow plot
kmeans = KMeans(n_clusters=4)
# Fitting the input data
kmeans = kmeans.fit(X)
# Obtaining the cluster labels
labels = kmeans.predict(X)
# Obtaining the Centroid values
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[8.34444215e+00 1.00059614e+04]
 [1.02725276e+02 1.68972439e+06]
 [2.22710427e+02 2.15013048e+05]
 [4.17985329e+02 7.18365309e+05]]
```

In [18]:
```
# Plotting the clusters
plt.scatter(data['revenue'], data['pageviews'], c= kmeans.labels_.astype(float
), s=50, alpha=0.5)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
```

Out[18]: <matplotlib.collections.PathCollection at 0x1e2eb213550>



In [107]:
```
## Importing the data for the black box method: one classsvm ...
## This data contains the computed "cpm"
## after manual computation of CPM as per the report instructions, I have obta
ined the CPM values, shown below
data = pd.read_csv("C:\\Users\\Snigdha.Cheekoty\\OneDrive - Serco\Desktop\\dai
ly_revenue123.csv")
```

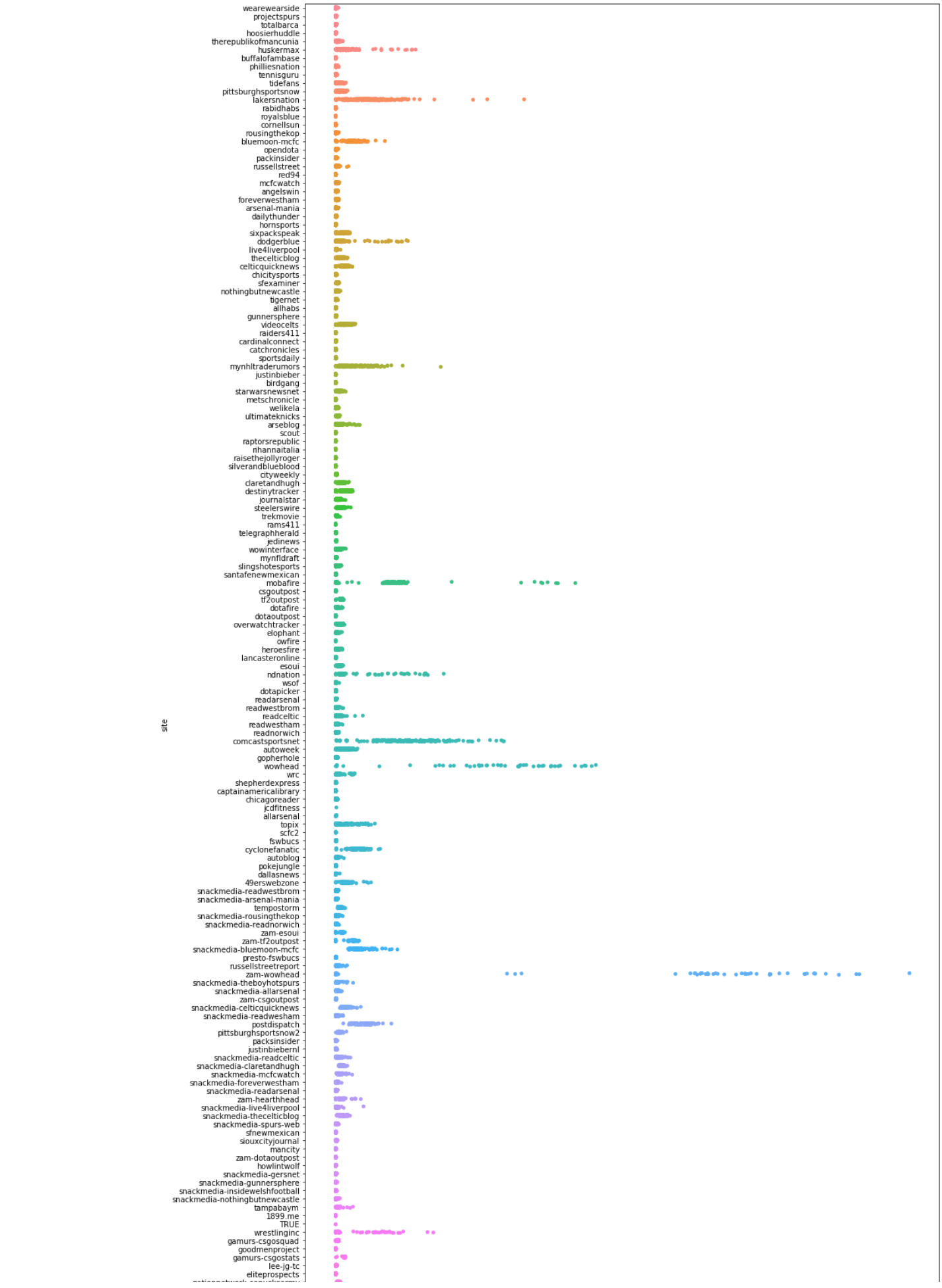
In [99]:
```
data.head(20)
```

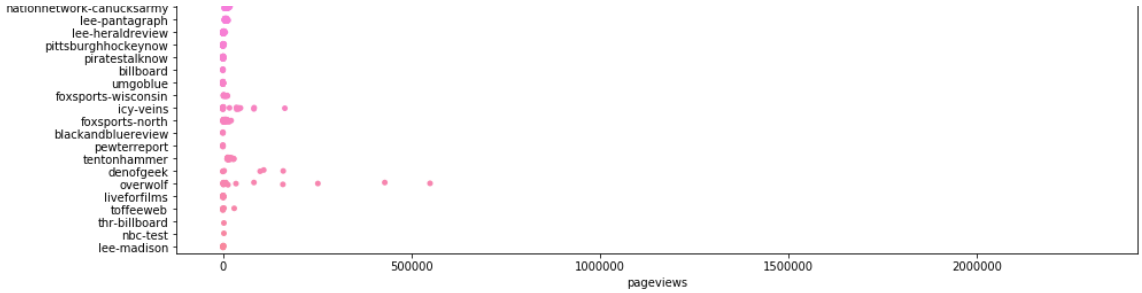
Out[99]:

	date	site	revenue	pageviews	cpm
0	13-07-2016	wearewearside	0.539353	389	1.386513
1	13-07-2016	projectspurs	3.588072	2353	1.524892
2	13-07-2016	totalbarca	5.130714	1228	4.178106
3	14-07-2016	projectspurs	4.264064	2762	1.543832
4	14-07-2016	totalbarca	2.280724	539	4.231400
5	14-07-2016	hoosierhuddle	0.219857	187	1.175705
6	14-07-2016	wearewearside	0.895627	638	1.403804
7	15-07-2016	hoosierhuddle	0.132319	173	0.764850
8	15-07-2016	wearewearside	0.064656	71	0.910647
9	15-07-2016	totalbarca	2.157703	785	2.748666
10	15-07-2016	projectspurs	2.853130	2844	1.003210
11	16-07-2016	projectspurs	1.683579	2257	0.745936
12	16-07-2016	wearewearside	0.390841	577	0.677367
13	16-07-2016	hoosierhuddle	0.068215	120	0.568462
14	16-07-2016	totalbarca	1.373814	672	2.044365
15	17-07-2016	projectspurs	0.841798	1180	0.713388
16	17-07-2016	wearewearside	0.507751	784	0.647642
17	17-07-2016	hoosierhuddle	0.081240	150	0.541601
18	17-07-2016	totalbarca	1.296636	663	1.955710
19	18-07-2016	therepublikofmancunia	3.052134	2524	1.209245

In [141]:
```
fig, ax = plt.subplots() # creating a figure
fig.set_size_inches(15,35)
sns.stripplot(data = data, y = "site", x = "pageviews")
#### Different sites and the correspondin pageviews (on a daily basis)
## You can see few abnormally high values pertaining to sesonality factors
```

Out[141]: <matplotlib.axes._subplots.AxesSubplot at 0x1e28f89cd30>


site
wearewearside
projectspurs
totalbarca
hoosierhuddle
therepublikofmancunia
huskermax
buffalofambase
philliesnation
tennisguru
tidefans
pittsburghsportsnow
lakersnation
rabidhabs
royalsblue
cornellsun
rousingthekop
bluemoon-mcfc
opendota
packinsider
russellstreet
red94
mcfcwatch
angelswin
foreverwestham
arsenal-mania
dailythunder
hornsports
sixpackspeak
dodgerblue
live4liverpool
thecelticblog
celticquicknews
chicitysports
sfexaminer
nothingbutnewcastle
tigernet
allhabs
gunnersphere
videocelts
raiders411
cardinalconnect
catchronicles
sportsdaily
mynhltraderumors
justinbieber
birdgang
starwarsnewsnet
metschronicle
welikela
ultimateknicks
arseblog
scout
raptorsrepublic
rihannaitalia
raisethejollyroger
silverandblueblood
cityweekly
claretandhugh
destinytracker
journalstar
steelerswire
trekmovie
rams411
telegraphherald
jedinews
wowinterface
mynfldraft
slingshotesports
santafenewmexican
mobafire
csgoutpost
tf2outpost
dotafire
dotaoutpost
overwatchtracker
elophant
owfire
heroesfire
lancasteronline
esoui
ndnation
wsof
dotapicker
readarsenal
readwestbrom
readceltic
readwestham
readnorwich
comcastsportsnet
autoweek
gopherhole
wowhead
wrc
shepherdexpress
captainamericalibrary
chicagoreader
jcdfitness
allarsenal
topix
scfc2
fswbucs
cyclonefanatic
autoblog
pokejungle
dallasnews
49erswebzone
snackmedia-readwestbrom
snackmedia-arsenal-mania
tempostorm
snackmedia-rousingthekop
snackmedia-readnorwich
zam-esoui
zam-tf2outpost
snackmedia-bluemoon-mcfc
presto-fswbucs
russellstreetreport
zam-wowhead
snackmedia-theboyhotspurs
snackmedia-allarsenal
zam-csgoutpost
snackmedia-celticquicknews
snackmedia-readwesham
postdispatch
pittsburghsportsnow2
packsinsider
justinbiebernl
snackmedia-readceltic
snackmedia-claretandhugh
snackmedia-mcfcwatch
snackmedia-foreverwestham
snackmedia-readarsenal
zam-hearthhead
snackmedia-live4liverpool
snackmedia-thecelticblog
snackmedia-spurs-web
sfnewmexican
siouxcityjournal
mancity
zam-dotaoutpost
howlintwolf
snackmedia-gersnet
snackmedia-gunnersphere
snackmedia-insidewelshfootball
snackmedia-nothingbutnewcastle
tampabaym
1899.me
TRUE
wrestlinginc
gamurs-csgosquad
goodmenproject
gamurs-csgostats
lee-jg-tc
eliteprospects



In [ ]:

In [86]:
```
## Importing the library for svm
from sklearn import svm
```

In [89]:
```
# Preparing the data
## Instead of using sampling, I have manually partioned the data into training
and test sets
## The trainset contains the records prior to 04/01/2017
## and the test set contains records after  04/01/2017
X = data[["revenue", "cpm"]]
train_feature = X.loc[0:4932, :]
train_feature = train_feature.drop('cpm', 1)
Y_1 = X.loc[4932:, "cpm"]
Y_2 = X['cpm']
```

In [90]:
```
# Creating test observations and features

X_test_1 = X.loc[4932:, :].drop('cpm',1)

X_test = X_test_1.append(X_test_2)
```

In [92]:
```
# Setting the hyperparameters for oneclass SVM
#Y_test is used to evaluate the model
oneclass = svm.OneClassSVM(kernel='linear', gamma=0.001, nu=0.95)
# Used various combination of hyperparameters like linear, rbf, poly, gamma-
Y_1 = X.loc[4320:, 'cpm']
Y_2 = X['cpm']
Y_test= Y_1.append(Y_2)
```

In [93]:
```
#training the model
oneclass.fit(train_feature)
```

Out[93]:
```
OneClassSVM(cache_size=200, coef0=0.0, degree=3, gamma=0.001, kernel='linea
r',
      max_iter=-1, nu=0.95, random_state=None, shrinking=True, tol=0.001,
      verbose=False)
```

In [94]:
```
# Testing the model on the validation set

fraud_pred = oneclass.predict(X_test)
```

In [95]:
```
# Check the number of outliers predicted by the algorithm

unique, counts = np.unique(fraud_pred, return_counts=True)
print (np.asarray((unique, counts)).T)
```

```
[[   -1 28909]
 [    1  1653]]
```

In [96]:
```
#Convert Y-test and fraud_pred to dataframe for ease of operation

Y_test= Y_test.to_frame()
Y_test=Y_test.reset_index()
fraud_pred = pd.DataFrame(fraud_pred)
fraud_pred= fraud_pred.rename(columns={0: 'prediction'})
```

In [97]:
```
fraud_pred[fraud_pred['prediction']==1]=0
fraud_pred[fraud_pred['prediction']==-1]=1
```

In [98]:
```
print(fraud_pred['prediction'].value_counts())
print(sum(fraud_pred['prediction'])/fraud_pred['prediction'].shape[0])
```

```
1    28909
0     1653
Name: prediction, dtype: int64
0.9459132255742425
```

In [ ]:

In [ ]: