CSPB-3022 FINAL PROJECT: Microsoft Malware Protection

Project Data Found here: https://www.kaggle.com/c/microsoft-malware-prediction/overview (https://www.kaggle.com/c/microsoft-malware-prediction/overview)

NAME= Daniel Shackelford

kaggle score: 0.57621

kaggle user id:User ID 5543986

kaggle username: danielshackelford

github repo: https://github.com/dash6877/CSPB3022_FINAL_PROJECT (https://github.com/dash6877/CSPB3022_FINAL_PROJECT)

Data Description:

The goal of this competition is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine. The telemetry data containing these properties and the machine infections was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender.

Each row in this dataset corresponds to a machine, uniquely identified by a MachineIdentifier. HasDetections is the ground truth and indicates that Malware was detected on the machine. Using the information and labels in train.csv, you must predict the value for HasDetections for each machine in test.csv.

The sampling methodology used to create this dataset was designed to meet certain business constraints, both in regards to user privacy as well as the time period during which the machine was running. Malware detection is inherently a time-series problem, but it is made complicated by the introduction of new machines, machines that come online and offline, machines that receive patches, machines that receive new operating systems, etc. While the dataset provided here has been roughly split by time, the complications and sampling requirements mentioned above may mean you may see imperfect agreement between your cross validation, public, and private scores! Additionally, this dataset is not representative of Microsoft customers' machines in the wild; it has been sampled to include a much larger proportion of malware machines.

Columns:

Unavailable or self-documenting column names are marked with an "NA".

MachineIdentifier - Individual machine ID
ProductName - Defender state information e.g. win8defender
EngineVersion - Defender state information e.g. 1.1.12603.0
AppVersion - Defender state information e.g. 4.9.10586.0
AvSigVersion - Defender state information e.g. 1.217.1014.0
IsBeta - Defender state information e.g. false
RtpStateBitfield - NA
IsSxsPassiveMode - NA
DefaultBrowsersIdentifier - ID for the machine's default browser
AVProductStatesIdentifier - ID for the specific configuration of a user's antivirus software
AVProductsInstalled - NA
AVProductsEnabled - NA
HasTpm - True if machine has tpm
CountryIdentifier - ID for the country the machine is located in
CityIdentifier - ID for the city the machine is located in
OrganizationIdentifier - ID for the organization the machine belongs in, organization ID is mapped to both specific companies and broad industries
GeoNameIdentifier - ID for the geographic region a machine is located in
LocaleEnglishNameIdentifier - English name of Locale ID of the current user
Platform - Calculates platform name (of OS related properties and processor property)
Processor - This is the process architecture of the installed operating system
OsVer - Version of the current operating system
OsBuild - Build of the current operating system
OsSuite - Product suite mask for the current operating system.
OsPlatformSubRelease - Returns the OS Platform sub-release (Windows Vista, Windows 7, Windows 8, TH1, TH2)
OsBuildLab - Build lab that generated the current OS. Example: 9600.17630.amd64fre.winblue_r7.150109-2022
SkuEdition - The goal of this feature is to use the Product Type defined in the MSDN to map to a 'SKU-Edition' name that is useful in population reporting. The valid Product Type are defined in %sdxroot%\data\windowseditions.xml. This API has been used since Vista and Server 2008, so there are many Product Types that do not apply to Windows 10. The 'SKU-Edition' is a string value that is in one of three classes of results. The design must hand each class.
IsProtected - This is a calculated field derived from the Spynet Report's AV Products field. Returns: a. TRUE if there is at least one active and up-to-date antivirus product running on this machine. b. FALSE if there is no active AV product on this machine, or if the AV is active, but is not receiving the latest updates. c. null if there are no Anti Virus Products in the report. Returns: Whether a machine is protected.
AutoSampleOptIn - This is the SubmitSamplesConsent value passed in from the service, available on CAMP 9+
PuaMode - Pua Enabled mode from the service
SMode - This field is set to true when the device is known to be in 'S Mode', as in, Windows 10 S mode, where only Microsoft Store apps can be installed
IeVerIdentifier - NA
SmartScreen - This is the SmartScreen enabled string value from registry. This is o

btained by checking in order, HKLM\SOFTWARE\Policies\Microsoft\Windows\System\SmartScreenEnabled and HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SmartScreenEnabled. If the value exists but is blank, the value "ExistsNotSet" is sent in telemetry.
Firewall - This attribute is true (1) for Windows 8.1 and above if windows firewall is enabled, as reported by the service.
UacLuaenable - This attribute reports whether or not the "administrator in Admin Approval Mode" user type is disabled or enabled in UAC. The value reported is obtained by reading the regkey HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA.
Census_MDC2FormFactor - A grouping based on a combination of Device Census level hardware characteristics. The logic used to define Form Factor is rooted in business and industry standards and aligns with how people think about their device. (Examples: Smartphone, Small Tablet, All in One, Convertible...)
Census_DeviceFamily - AKA DeviceClass. Indicates the type of device that an edition of the OS is intended for. Example values: Windows.Desktop, Windows.Mobile, and iOS.Phone
Census_OEMNameIdentifier - NA
Census_OEMModelIdentifier - NA
Census_ProcessorCoreCount - Number of logical cores in the processor
Census_ProcessorManufacturerIdentifier - NA
Census_ProcessorModelIdentifier - NA
Census_ProcessorClass - A classification of processors into high/medium/low. Initially used for Pricing Level SKU. No longer maintained and updated
Census_PrimaryDiskTotalCapacity - Amount of disk space on primary disk of the machine in MB
Census_PrimaryDiskTypeName - Friendly name of Primary Disk Type - HDD or SSD
Census_SystemVolumeTotalCapacity - The size of the partition that the System volume is installed on in MB
Census_HasOpticalDiskDrive - True indicates that the machine has an optical disk drive (CD/DVD)
Census_TotalPhysicalRAM - Retrieves the physical RAM in MB
Census_ChassisTypeName - Retrieves a numeric representation of what type of chassis the machine has. A value of 0 means xx
Census_InternalPrimaryDiagonalDisplaySizeInInches - Retrieves the physical diagonal length in inches of the primary display
Census_InternalPrimaryDisplayResolutionHorizontal - Retrieves the number of pixels in the horizontal direction of the internal display.
Census_InternalPrimaryDisplayResolutionVertical - Retrieves the number of pixels in the vertical direction of the internal display
Census_PowerPlatformRoleName - Indicates the OEM preferred power management profile. This value helps identify the basic form factor of the device
Census_InternalBatteryType - NA
Census_InternalBatteryNumberOfCharges - NA
Census_OSVersion - Numeric OS version Example - 10.0.10130.0
Census_OSArchitecture - Architecture on which the OS is based. Derived from OSVersi

In [1]:

```
%matplotlib inline
import numpy as np
import scipy as sp
import scipy.stats as stats
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import patsy
import sklearn
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
import seaborn as sns

import scipy.stats
import sklearn.linear_model
import sklearn.discriminant_analysis
import sklearn.preprocessing
import sklearn.model_selection
import sklearn.neighbors
```

In [2]:
```
#load objects as categories
#Switch bianry values to int8
#binary with missing values are imported as float16
#change 64 bit floats to 32 or 16 if possible.

dtypes = {
        'MachineIdentifier':                                    'category',
        'ProductName':                                          'category',
        'EngineVersion':                                        'category',
        'AppVersion':                                           'category',
        'AvSigVersion':                                         'category',
        'IsBeta':                                               'int8',
        'RtpStateBitfield':                                     'float16',
        'IsSxsPassiveMode':                                     'int8',
        'DefaultBrowsersIdentifier':                            'float32',
        'AVProductStatesIdentifier':                            'float32',
        'AVProductsInstalled':                                  'float16',
        'AVProductsEnabled':                                    'float16',
        'HasTpm':                                               'int8',
        'CountryIdentifier':                                    'int16',
        'CityIdentifier':                                       'float32',
        'OrganizationIdentifier':                               'float16',
        'GeoNameIdentifier':                                    'float16',
        'LocaleEnglishNameIdentifier':                          'int16',
        'Platform':                                             'category',
        'Processor':                                            'category',
        'OsVer':                                                'category',
        'OsBuild':                                              'int16',
        'OsSuite':                                              'int16',
        'OsPlatformSubRelease':                                 'category',
        'OsBuildLab':                                           'category',
        'SkuEdition':                                           'category',
        'IsProtected':                                          'float16',
        'AutoSampleOptIn':                                      'int8',
        'PuaMode':                                              'category',
        'SMode':                                                'float16',
        'IeVerIdentifier':                                      'float16',
        'SmartScreen':                                          'category',
        'Firewall':                                             'float16',
        'UacLuaenable':                                         'float32',
        'UacLuaenable':                                         'float64', # w
as 'float32'
        'Census_MDC2FormFactor':                                'category',
        'Census_DeviceFamily':                                  'category',
        'Census_OEMNameIdentifier':                             'float32', # w
as 'float16'
        'Census_OEMModelIdentifier':                            'float32',
        'Census_ProcessorCoreCount':                            'float16',
        'Census_ProcessorManufacturerIdentifier':               'float16',
        'Census_ProcessorModelIdentifier':                      'float32', # w
as 'float16'
        'Census_ProcessorClass':                                'category',
        'Census_PrimaryDiskTotalCapacity':                      'float64', # w
as 'float32'
        'Census_PrimaryDiskTypeName':                           'category',
        'Census_SystemVolumeTotalCapacity':                     'float64', # w
```

```
as 'float32'
        'Census_HasOpticalDiskDrive':                           'int8',
        'Census_TotalPhysicalRAM':                              'float32',
        'Census_ChassisTypeName':                               'category',
        'Census_InternalPrimaryDiagonalDisplaySizeInInches':    'float32', # w
as 'float16'
        'Census_InternalPrimaryDisplayResolutionHorizontal':    'float32', # w
as 'float16'
        'Census_InternalPrimaryDisplayResolutionVertical':      'float32', # w
as 'float16'
        'Census_PowerPlatformRoleName':                         'category',
        'Census_InternalBatteryType':                           'category',
        'Census_InternalBatteryNumberOfCharges':                'float64', # w
as 'float32'
        'Census_OSVersion':                                     'category',
        'Census_OSArchitecture':                                'category',
        'Census_OSBranch':                                      'category',
        'Census_OSBuildNumber':                                 'int16',
        'Census_OSBuildRevision':                               'int32',
        'Census_OSEdition':                                     'category',
        'Census_OSSkuName':                                     'category',
        'Census_OSInstallTypeName':                             'category',
        'Census_OSInstallLanguageIdentifier':                   'float16',
        'Census_OSUILocaleIdentifier':                          'int16',
        'Census_OSWUAutoUpdateOptionsName':                     'category',
        'Census_IsPortableOperatingSystem':                     'int8',
        'Census_GenuineStateName':                              'category',
        'Census_ActivationChannel':                             'category',
        'Census_IsFlightingInternal':                           'float16',
        'Census_IsFlightsDisabled':                             'float16',
        'Census_FlightRing':                                    'category',
        'Census_ThresholdOptIn':                                'float16',
        'Census_FirmwareManufacturerIdentifier':                'float16',
        'Census_FirmwareVersionIdentifier':                     'float32',
        'Census_IsSecureBootEnabled':                           'int8',
        'Census_IsWIMBootEnabled':                              'float16',
        'Census_IsVirtualDevice':                               'float16',
        'Census_IsTouchEnabled':                                'int8',
        'Census_IsPenCapable':                                  'int8',
        'Census_IsAlwaysOnAlwaysConnectedCapable':              'float16',
        'Wdft_IsGamer':                                         'float16',
        'Wdft_RegionIdentifier':                                'float16',
        'HasDetections':                                        'int8'
        }



%time train_data=pd.read_csv(r'C:\Users\Dshac\OneDrive\Documents\CSPB 3022\Pro
ject\microsoft-malware-prediction\train.csv', low_memory=False)
```

```
Wall time: 1min 46s
```

In [3]:
```
train_data=train_data.astype(dtypes)
```

In [4]:
```
train_data.dtypes
```

Out[4]:
```
MachineIdentifier                             category
ProductName                                   category
EngineVersion                                 category
AppVersion                                    category
AvSigVersion                                  category
                                                 ...
Census_IsPenCapable                              int8
Census_IsAlwaysOnAlwaysConnectedCapable       float16
Wdft_IsGamer                                  float16
Wdft_RegionIdentifier                         float16
HasDetections                                    int8
Length: 83, dtype: object
```

In [5]:
```
#first look for columns that are missing most of their data, higher=more missi
ng
(train_data.isnull().sum()/train_data.shape[0]).sort_values(ascending=False)
```

Out[5]:
```
PuaMode                        0.999741
Census_ProcessorClass          0.995894
DefaultBrowsersIdentifier      0.951416
Census_IsFlightingInternal     0.830440
Census_InternalBatteryType     0.710468
                                   ...
Census_OSVersion               0.000000
Census_HasOpticalDiskDrive     0.000000
Census_DeviceFamily            0.000000
Census_MDC2FormFactor          0.000000
MachineIdentifier              0.000000
Length: 83, dtype: float64
```

In [6]:
```
#Two columns can be seen to have mroe than 99% missing data...
drop_features=[] #create blnak array to hold features to be dropped
drop_features.append('PuaMode')
drop_features.append('Census_ProcessorClass')
print(drop_features)
```

```
['PuaMode', 'Census_ProcessorClass']
```

In [7]:
```
pd.options.display.float_format = '{:,.4f}'.format
sk_df = pd.DataFrame([{'column': c, 'uniq': train_data[c].nunique(), 'skewnes
s': train_data[c].value_counts(normalize=True).values[0] * 100} for c in train
_data.columns])
sk_df = sk_df.sort_values('skewness', ascending=False)
```

In [8]:
```
sk_df.head(40)
#This data means that there are 12 categories that have a majority
#category that covers more than 99% of occurences, making them not usable.
```

Out[8]:

	column	uniq	skewness
75	Census_IsWIMBootEnabled	2	100.0000
5	IsBeta	2	99.9992
69	Census_IsFlightsDisabled	2	99.9990
68	Census_IsFlightingInternal	2	99.9986
27	AutoSampleOptIn	2	99.9971
71	Census_ThresholdOptIn	2	99.9749
29	SMode	2	99.9537
65	Census_IsPortableOperatingSystem	2	99.9455
28	PuaMode	2	99.9134
35	Census_DeviceFamily	3	99.8383
33	UacLuaenable	11	99.3925
76	Census_IsVirtualDevice	2	99.2961
1	ProductName	6	98.9356
12	HasTpm	2	98.7971
7	IsSxsPassiveMode	2	98.2666
32	Firewall	2	97.8583
11	AVProductsEnabled	6	97.3984
6	RtpStateBitfield	7	97.3262
20	OsVer	58	96.7613
18	Platform	4	96.6063
78	Census_IsPenCapable	2	96.1929
26	IsProtected	2	94.5624
79	Census_IsAlwaysOnAlwaysConnectedCapable	2	94.2581
70	Census_FlightRing	10	93.6580
45	Census_HasOpticalDiskDrive	2	92.2813
55	Census_OSArchitecture	3	90.8580
19	Processor	3	90.8530
66	Census_GenuineStateName	5	88.2992
39	Census_ProcessorManufacturerIdentifier	7	88.2789
77	Census_IsTouchEnabled	2	87.4457
52	Census_InternalBatteryType	78	78.5216
31	SmartScreen	21	75.1363
80	Wdft_IsGamer	2	71.6421
10	AVProductsInstalled	8	69.8786
51	Census_PowerPlatformRoleName	10	69.3040

	column	uniq	skewness
15	OrganizationIdentifier	49	68.0143
9	AVProductStatesIdentifier	28970	65.5531
43	Census_PrimaryDiskTypeName	4	65.1817
34	Census_MDC2FormFactor	13	64.1521
22	OsSuite	14	62.3289

In [9]:
```
drop_features.append('Census_IsWIMBootEnabled')
drop_features.append('IsBeta')
drop_features.append('Census_IsFlightsDisabled')
drop_features.append('Census_IsFlightingInternal')
drop_features.append('AutoSampleOptIn')
drop_features.append('Census_ThresholdOptIn')
drop_features.append('SMode')
drop_features.append('Census_IsPortableOperatingSystem')
drop_features.append('Census_DeviceFamily')
drop_features.append('UacLuaenable')
drop_features.append('Census_IsVirtualDevice')
drop_features
#drop all values that are not duplicates of the first missing data test
```

Out[9]:
```
['PuaMode',
 'Census_ProcessorClass',
 'Census_IsWIMBootEnabled',
 'IsBeta',
 'Census_IsFlightsDisabled',
 'Census_IsFlightingInternal',
 'AutoSampleOptIn',
 'Census_ThresholdOptIn',
 'SMode',
 'Census_IsPortableOperatingSystem',
 'Census_DeviceFamily',
 'UacLuaenable',
 'Census_IsVirtualDevice']
```

In [10]:
```
train_data.drop(drop_features,axis=1,inplace=True)
```

In [11]:
```
#check for null data
train_null=train_data.isnull().sum()
train_null=train_null/train_data.shape[0]
train_null[train_null > 0.1]
```

Out[11]:
```
DefaultBrowsersIdentifier    0.9514
OrganizationIdentifier       0.3084
SmartScreen                  0.3561
Census_InternalBatteryType   0.7105
dtype: float64
```

In [12]:
```
train_data.DefaultBrowsersIdentifier.value_counts().head(5)
```

Out[12]:
```
239.0000      46056
3,195.0000    42692
1,632.0000    28751
3,176.0000    24220
146.0000      20756
Name: DefaultBrowsersIdentifier, dtype: int64
```

In [13]:
```
train_data.DefaultBrowsersIdentifier.fillna(0, inplace=True)
```

In [14]:
```
train_data.SmartScreen.value_counts()
```

Out[14]:
```
RequireAdmin    4316183
ExistsNotSet    1046183
Off              186553
Warn             135483
Prompt            34533
Block             22533
off                1350
On                  731
&#x02;              416
&#x01;              335
on                  147
requireadmin         10
OFF                   4
0                     3
Promt                 2
Enabled               1
prompt                1
00000000              1
&#x03;                1
requireAdmin          1
warn                  1
Name: SmartScreen, dtype: int64
```

In [15]:
```
trans_dict = {
    'off': 'Off', '&#x02;': '2', '&#x01;': '1', 'on': 'On', 'requireadmin': 'RequireAdmin', 'OFF': 'Off',
    'Promt': 'Prompt', 'requireAdmin': 'RequireAdmin', 'prompt': 'Prompt', 'warn': 'Warn',
    '00000000': '0', '&#x03;': '3', np.nan: 'NoExist'
}
train_data.replace({'SmartScreen': trans_dict}, inplace=True)
```

In [16]:
```
train_data.SmartScreen.isnull().sum()
```

Out[16]:
```
0
```

In [17]:
```
train_data.OrganizationIdentifier.value_counts()
```

Out[17]:
```
27.0000    4196457
18.0000    1764175
48.0000      63845
50.0000      45502
11.0000      19436
37.0000      19398
49.0000      13627
46.0000      10974
14.0000       4713
32.0000       4045
36.0000       3909
52.0000       3043
33.0000       2896
2.0000        2595
5.0000        1990
40.0000       1648
28.0000       1591
4.0000        1385
10.0000       1083
51.0000        917
20.0000        915
1.0000         893
8.0000         723
22.0000        418
39.0000        413
6.0000         412
31.0000        398
21.0000        397
47.0000        385
3.0000         331
16.0000        242
19.0000        172
26.0000        160
44.0000        150
29.0000        135
42.0000        132
7.0000          98
41.0000         77
45.0000         73
30.0000         64
43.0000         60
35.0000         32
23.0000         20
15.0000         13
25.0000         12
12.0000          7
34.0000          2
38.0000          1
17.0000          1
Name: OrganizationIdentifier, dtype: int64
```

In [18]:
```
train_data.replace({'OrganizationIdentifier': {np.nan: 0}}, inplace=True)
```

In [19]:
```
pd.options.display.max_rows = 99
train_data.Census_InternalBatteryType.value_counts()
```

Out[19]:

lion	2028256
li-i	245617
#	183998
lip	62099
liio	32635
li p	8383
li	6708
nimh	4614
real	2744
bq20	2302
pbac	2274
vbox	1454
unkn	533
lgi0	399
lipo	198
lhp0	182
4cel	170
lipp	83
ithi	79
batt	60
ram	35
bad	33
virt	33
pad0	22
lit	16
ca48	16
a132	10
ots0	9
lai0	8
ÿÿÿÿ	8
lio	5
asmb	4
4lio	4
li-p	4
□lio	4
icp3	3
0x0b	3
lgs0	3
lhpo	2
3ion	2
a138	2
a140	2
5nm1	2
h00j	2
#TAB#	1
0ts0	1
sail	1
2337	1
@i□□	1
3500	1
sams	1
pbso	1
4ion	1
pa50	1
□···	1
6ion	1
í□-i	1

```
□ion              1
8                 1
p-sn              1
a130              1
÷ÿóö              1
lgl0              1
li□               1
li-h              1
l□                1
l□&#TAB#          1
li-l              1
li?               1
h4°s              1
d                 1
cl53              1
lilo              1
li□o              1
liÿÿ              1
lp                1
lÿÿÿ              1
□ip               1
Name: Census_InternalBatteryType, dtype: int64
```

In [20]:
```
trans_dict = {
    '□```': 'unknown', 'unkn': 'unknown', np.nan: 'unknown'
}
train_data.replace({'Census_InternalBatteryType': trans_dict}, inplace=True)
```

In [21]:
```
train_data.shape
```

Out[21]:
```
(8921483, 70)
```

In [22]:
```
train_data.dropna(inplace=True)
train_data.shape
```

Out[22]:
```
(7667789, 70)
```

In [23]:
```
train_data.drop('MachineIdentifier', axis=1, inplace=True)
```

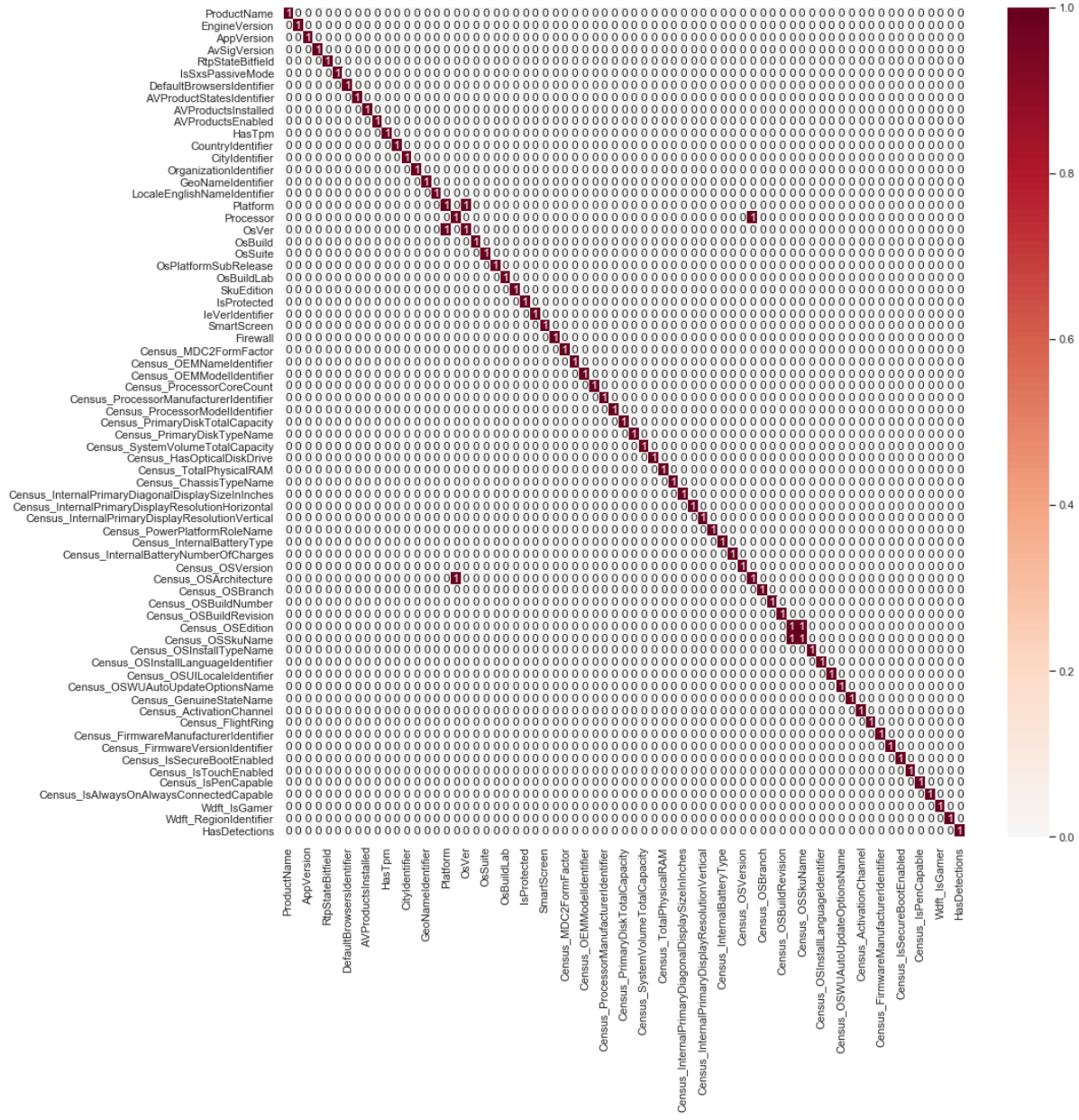
In [24]:
```
train_data['SmartScreen'] = train_data.SmartScreen.astype('category')
train_data['Census_InternalBatteryType'] = train_data.Census_InternalBatteryTy
pe.astype('category')

cate_cols = train_data.select_dtypes(include='category').columns.tolist()

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for col in cate_cols:
    train_data[col] = le.fit_transform(train_data[col])
```

In [ ]:
```

```

In [25]:
```
def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the data type
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        else:
            df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df

%time
train_data = reduce_mem_usage(train_data)
```

```
Wall time: 0 ns
Memory usage of dataframe is 1703.83 MB
Memory usage after optimization is: 965.26 MB
Decreased by 43.3%
```

In [26]:
```
# show a plot of all correlations in remaining data.
corr = train_data.corr()
high_corr = (corr >= 0.99).astype('uint8')
plt.figure(figsize=(15,15))
sns.heatmap(high_corr, cmap='RdBu_r', annot=True, center=0.0)
plt.show()
```

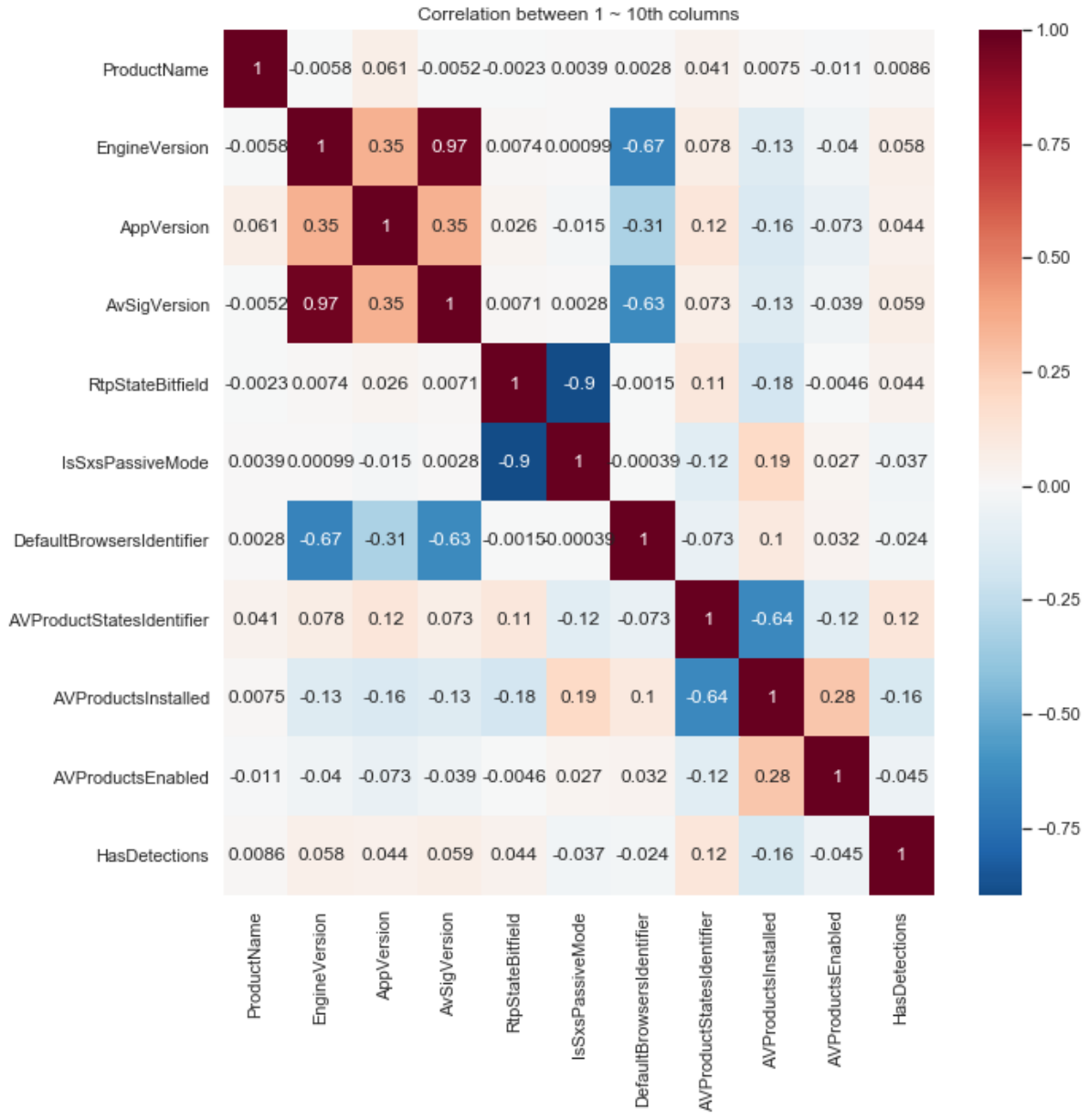


In [27]:
```
cols = train_data.columns.tolist()
print(cols[:10])
```

```
['ProductName', 'EngineVersion', 'AppVersion', 'AvSigVersion', 'RtpStateBitfi
eld', 'IsSxsPassiveMode', 'DefaultBrowsersIdentifier', 'AVProductStatesIdenti
fier', 'AVProductsInstalled', 'AVProductsEnabled']
```

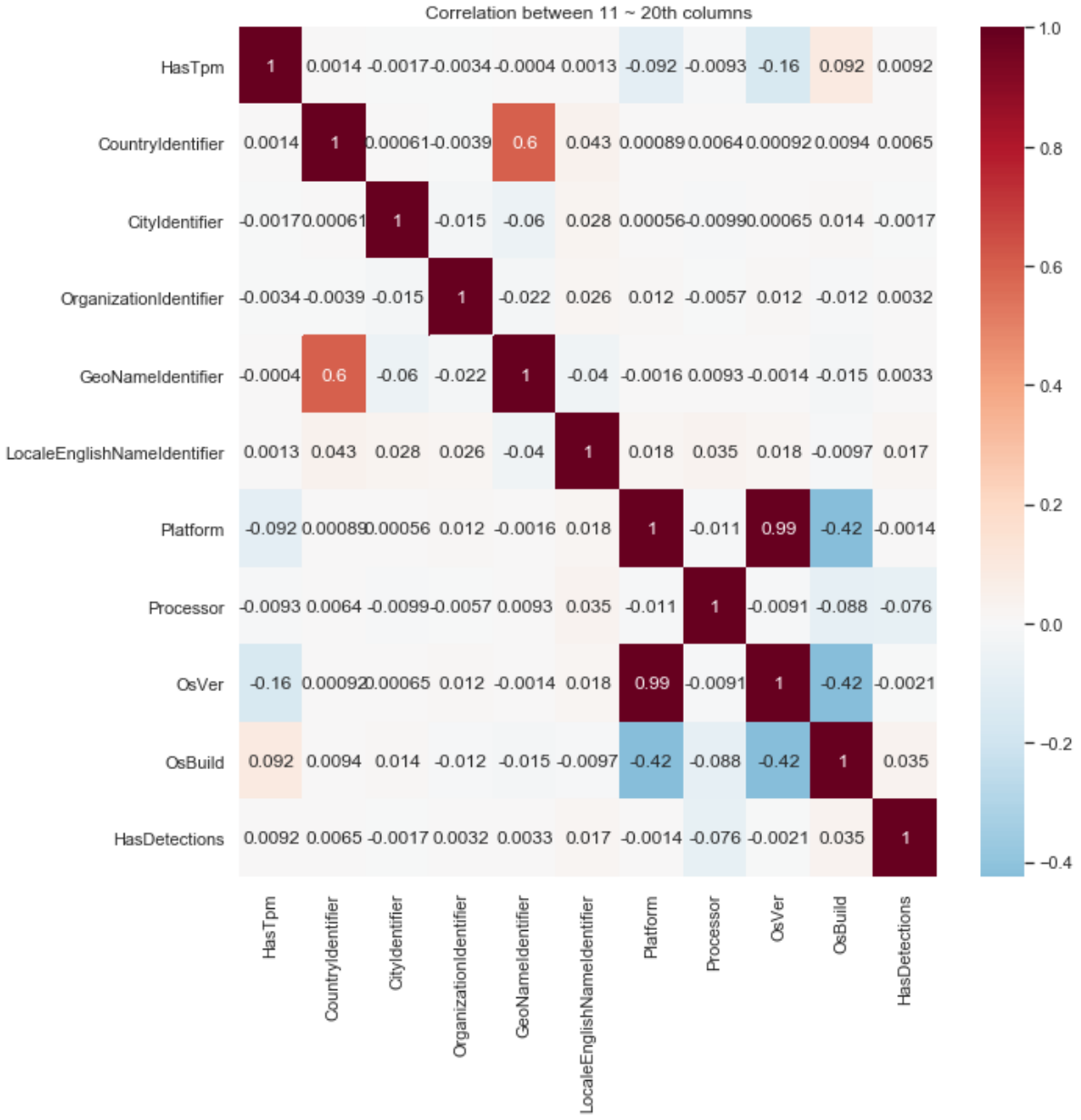
In [28]:
```
plt.figure(figsize=(10,10))
co_cols = cols[:10]
print(co_cols)
target_train=train_data['HasDetections']
co_cols.append('HasDetections')
sns.heatmap(train_data[co_cols].corr(), cmap='RdBu_r', annot=True, center=0.0)
plt.title('Correlation between 1 ~ 10th columns')
plt.show()
```

```
['ProductName', 'EngineVersion', 'AppVersion', 'AvSigVersion', 'RtpStateBitfi
eld', 'IsSxsPassiveMode', 'DefaultBrowsersIdentifier', 'AVProductStatesIdenti
fier', 'AVProductsInstalled', 'AVProductsEnabled']
```



In [29]:
```
corr_remove = []
```

In [30]:
```
co_cols = cols[10:20]
co_cols.append('HasDetections')
plt.figure(figsize=(10,10))
sns.heatmap(train_data[co_cols].corr(), cmap='RdBu_r', annot=True, center=0.0)
plt.title('Correlation between 11 ~ 20th columns')
plt.show()
```



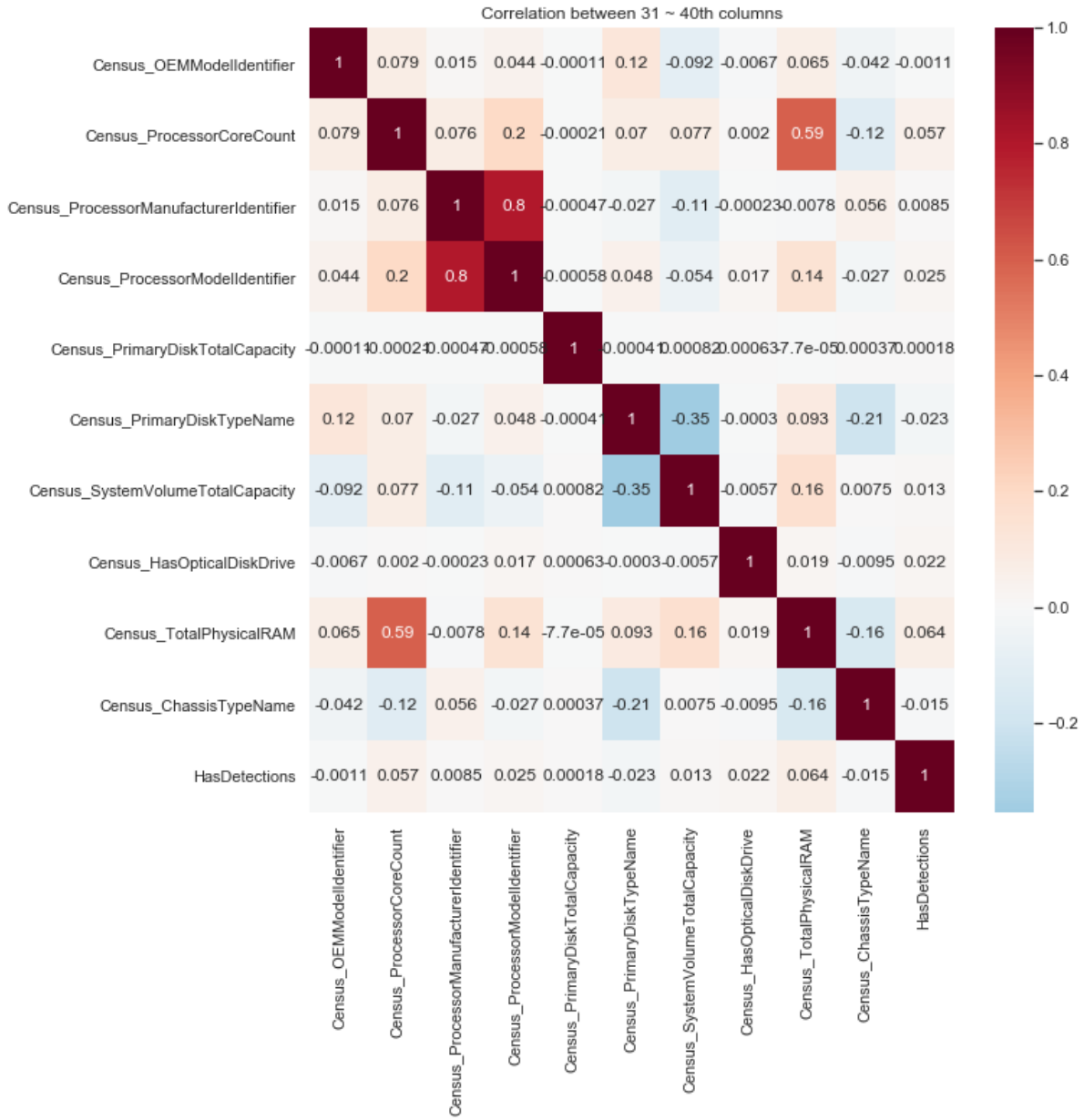
In [31]:
```
#which feature has less unique values?
print(train_data.Platform.nunique())
print(train_data.OsVer.nunique())
```

```
3
45
```

In [32]:
```
corr_remove.append('Platform')
```

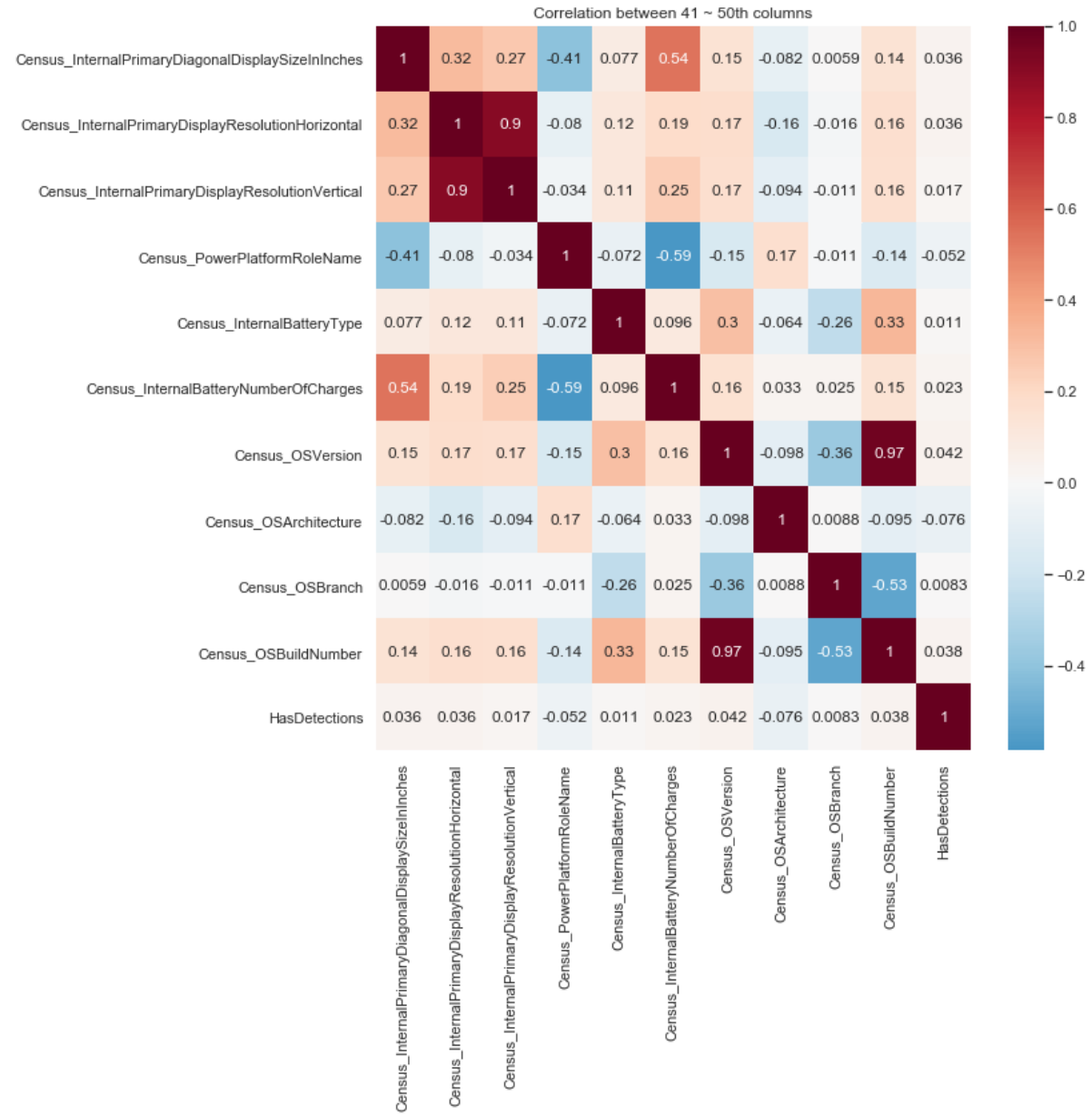
In [33]:

```
co_cols = cols[30:40]
co_cols.append('HasDetections')
plt.figure(figsize=(10,10))
sns.heatmap(train_data[co_cols].corr(), cmap='RdBu_r', annot=True, center=0.0)
plt.title('Correlation between 31 ~ 40th columns')
plt.show()
```

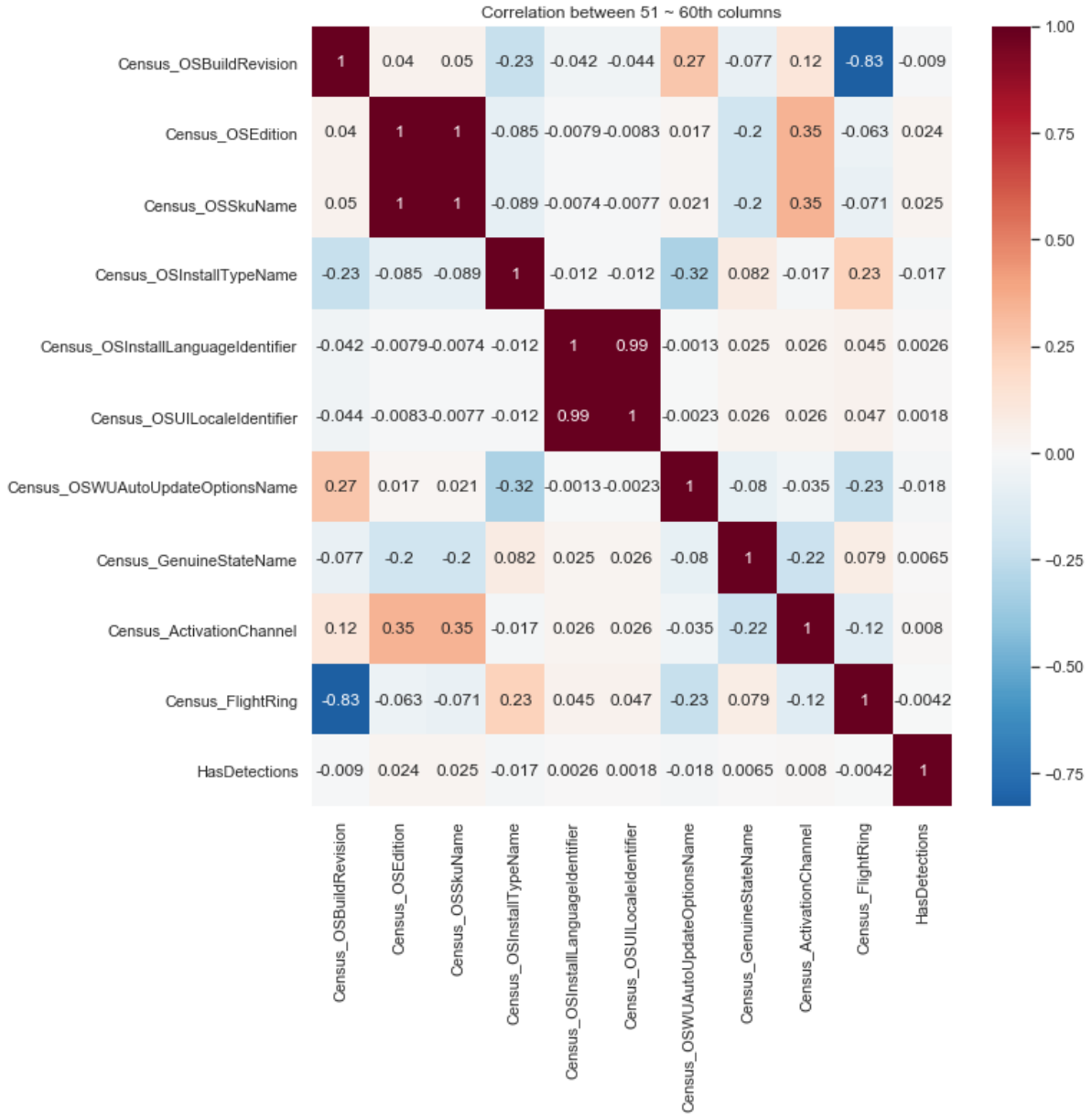


In [34]:

```
co_cols = cols[40:50]
co_cols.append('HasDetections')
plt.figure(figsize=(10,10))
sns.heatmap(train_data[co_cols].corr(), cmap='RdBu_r', annot=True, center=0.0)
plt.title('Correlation between 41 ~ 50th columns')
plt.show()
```



In [35]:
```
co_cols = cols[50:60]
co_cols.append('HasDetections')
plt.figure(figsize=(10,10))
sns.heatmap(train_data[co_cols].corr(), cmap='RdBu_r', annot=True, center=0.0)
plt.title('Correlation between 51 ~ 60th columns')
plt.show()
```

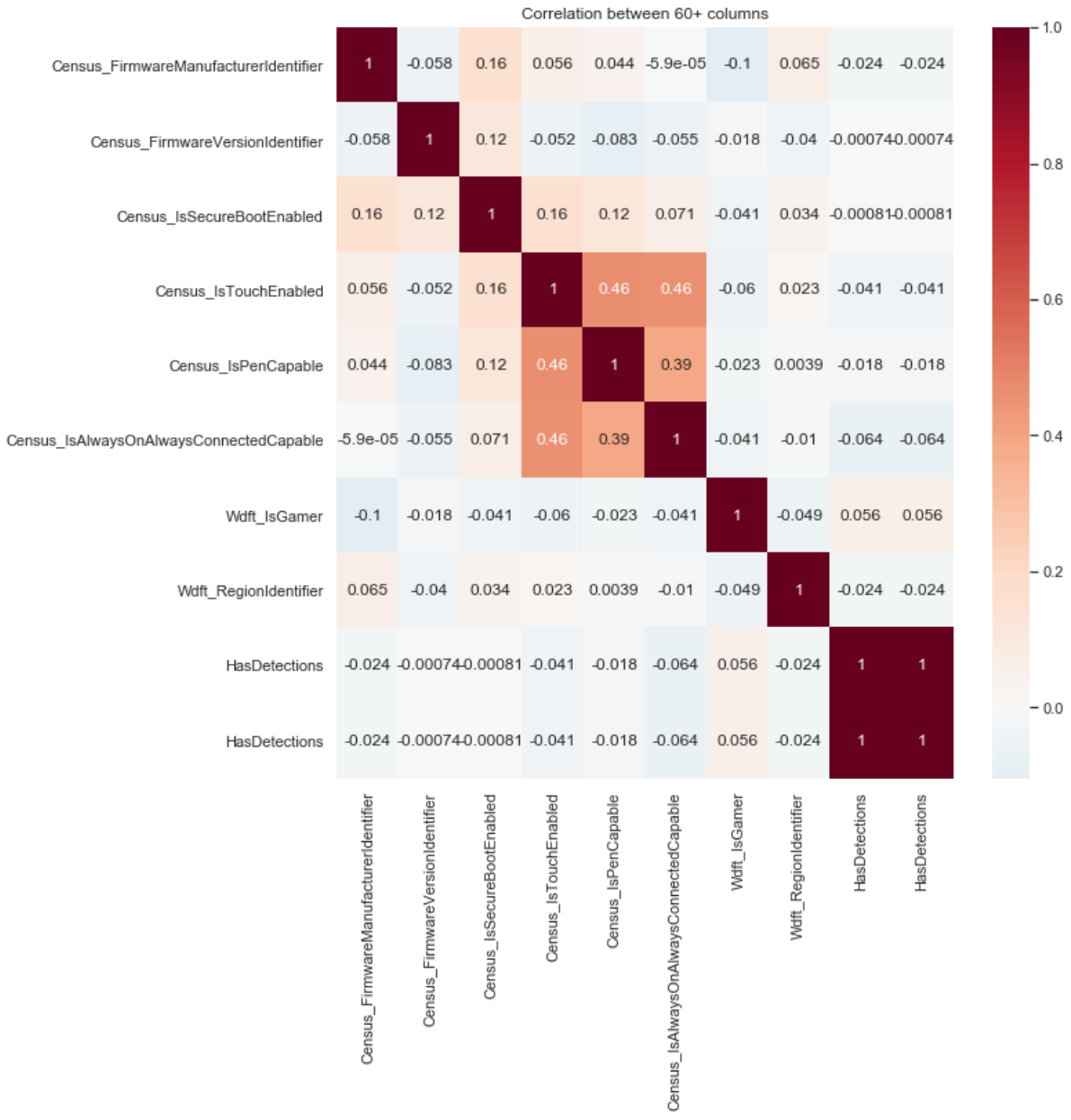


In [36]:
```
print(train_data.Census_OSEdition.nunique())
print(train_data.Census_OSSkuName.nunique())
print(train_data.Census_OSInstallLanguageIdentifier.nunique())
print(train_data.Census_OSUILocaleIdentifier.nunique())
```

```
29
25
39
144
```

In [37]:
```
corr_remove.append('Census_OSSkuName')
corr_remove.append('Census_OSInstallLanguageIdentifier')
```

In [38]:
```
co_cols = cols[60:]
co_cols.append('HasDetections')
plt.figure(figsize=(10,10))
sns.heatmap(train_data[co_cols].corr(), cmap='RdBu_r', annot=True, center=0.0)
plt.title('Correlation between 60+ columns')
plt.show()
```

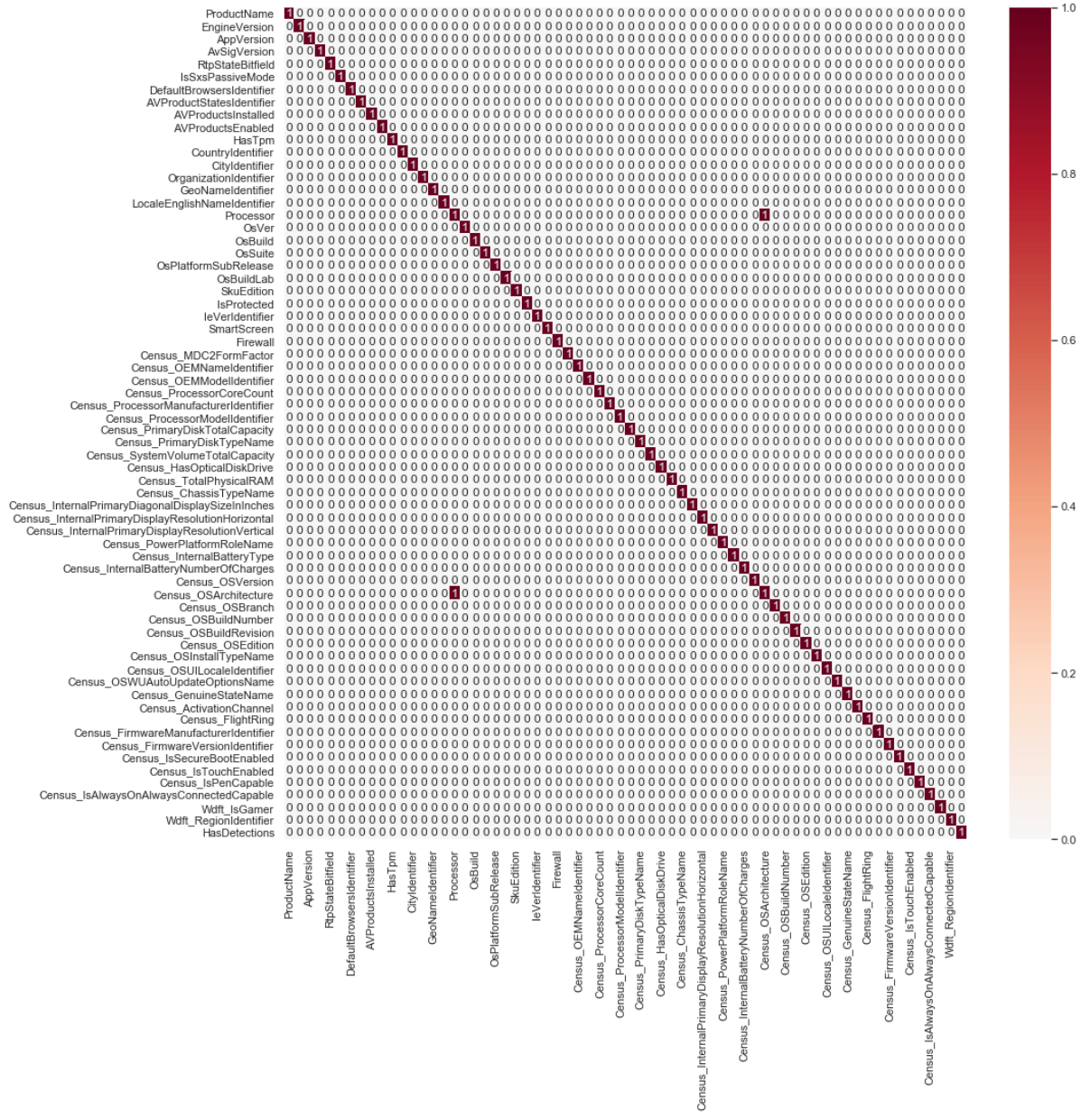


In [39]:
```
corr_remove
```

Out[39]:
```
['Platform', 'Census_OSSkuName', 'Census_OSInstallLanguageIdentifier']
```

In [40]:
```
train_data.drop(corr_remove, axis=1, inplace=True)
```

In [41]:
```
corr = train_data.corr()
high_corr = (corr >= 0.99).astype('uint8')
plt.figure(figsize=(15,15))
sns.heatmap(high_corr, cmap='RdBu_r', annot=True, center=0.0)
plt.show()
```



In [42]:
```
print(train_data.Census_OSArchitecture.nunique())
print(train_data.Processor.nunique())
```

```
3
3
```

In [43]:
```
train_data[['Census_OSArchitecture', 'Processor', 'HasDetections']].corr()
```

Out[43]:

	Census_OSArchitecture	Processor	HasDetections
Census_OSArchitecture	1.0000	0.9951	-0.0758
Processor	0.9951	1.0000	-0.0758
HasDetections	-0.0758	-0.0758	1.0000

In [44]:
```
corr_remove.append('Processor')
```

In [45]:
```
drop_features.extend(corr_remove)
print("Number of features that can be dropepd: ",len(drop_features))
drop_features
```

```
Number of features that can be dropepd:  17
```

Out[45]:
```
['PuaMode',
 'Census_ProcessorClass',
 'Census_IsWIMBootEnabled',
 'IsBeta',
 'Census_IsFlightsDisabled',
 'Census_IsFlightingInternal',
 'AutoSampleOptIn',
 'Census_ThresholdOptIn',
 'SMode',
 'Census_IsPortableOperatingSystem',
 'Census_DeviceFamily',
 'UacLuaenable',
 'Census_IsVirtualDevice',
 'Platform',
 'Census_OSSkuName',
 'Census_OSInstallLanguageIdentifier',
 'Processor']
```

In [46]:
```
train_opt=train_data
train_opt.drop('Processor', axis=1, inplace=True)
```

In [47]:
```
train_opt.head(30)
train_opt.dtypes
```

Out[47]:

ProductName	int8
EngineVersion	int8
AppVersion	int8
AvSigVersion	int16
RtpStateBitfield	float16
IsSxsPassiveMode	int8
DefaultBrowsersIdentifier	float16
AVProductStatesIdentifier	float32
AVProductsInstalled	float16
AVProductsEnabled	float16
HasTpm	int8
CountryIdentifier	int16
CityIdentifier	float32
OrganizationIdentifier	float16
GeoNameIdentifier	float16
LocaleEnglishNameIdentifier	int16
OsVer	int8
OsBuild	int16
OsSuite	int16
OsPlatformSubRelease	int8
OsBuildLab	int16
SkuEdition	int8
IsProtected	float16
IeVerIdentifier	float16
SmartScreen	int8
Firewall	float16
Census_MDC2FormFactor	int8
Census_OEMNameIdentifier	float16
Census_OEMModelIdentifier	float32
Census_ProcessorCoreCount	float16
Census_ProcessorManufacturerIdentifier	float16
Census_ProcessorModelIdentifier	float16
Census_PrimaryDiskTotalCapacity	float32
Census_PrimaryDiskTypeName	int8
Census_SystemVolumeTotalCapacity	float32
Census_HasOpticalDiskDrive	int8
Census_TotalPhysicalRAM	float32
Census_ChassisTypeName	int8
Census_InternalPrimaryDiagonalDisplaySizeInInches	float16
Census_InternalPrimaryDisplayResolutionHorizontal	float16
Census_InternalPrimaryDisplayResolutionVertical	float16
Census_PowerPlatformRoleName	int8
Census_InternalBatteryType	int8
Census_InternalBatteryNumberOfCharges	float32
Census_OSVersion	int16
Census_OSArchitecture	int8
Census_OSBranch	int8
Census_OSBuildNumber	int16
Census_OSBuildRevision	int16
Census_OSEdition	int8
Census_OSInstallTypeName	int8
Census_OSUILocaleIdentifier	int16
Census_OSWUAutoUpdateOptionsName	int8
Census_GenuineStateName	int8
Census_ActivationChannel	int8
Census_FlightRing	int8
Census_FirmwareManufacturerIdentifier	float16

```
Census_FirmwareVersionIdentifier                    float32
Census_IsSecureBootEnabled                             int8
Census_IsTouchEnabled                                  int8
Census_IsPenCapable                                    int8
Census_IsAlwaysOnAlwaysConnectedCapable             float16
Wdft_IsGamer                                        float16
Wdft_RegionIdentifier                               float16
HasDetections                                          int8
dtype: object
```

In [48]:
```
train_opt.shape
types=train_opt.dtypes
column=train_opt.columns
print(types)
print(column)
```

ProductName	int8
EngineVersion	int8
AppVersion	int8
AvSigVersion	int16
RtpStateBitfield	float16
IsSxsPassiveMode	int8
DefaultBrowsersIdentifier	float16
AVProductStatesIdentifier	float32
AVProductsInstalled	float16
AVProductsEnabled	float16
HasTpm	int8
CountryIdentifier	int16
CityIdentifier	float32
OrganizationIdentifier	float16
GeoNameIdentifier	float16
LocaleEnglishNameIdentifier	int16
OsVer	int8
OsBuild	int16
OsSuite	int16
OsPlatformSubRelease	int8
OsBuildLab	int16
SkuEdition	int8
IsProtected	float16
IeVerIdentifier	float16
SmartScreen	int8
Firewall	float16
Census_MDC2FormFactor	int8
Census_OEMNameIdentifier	float16
Census_OEMModelIdentifier	float32
Census_ProcessorCoreCount	float16
Census_ProcessorManufacturerIdentifier	float16
Census_ProcessorModelIdentifier	float16
Census_PrimaryDiskTotalCapacity	float32
Census_PrimaryDiskTypeName	int8
Census_SystemVolumeTotalCapacity	float32
Census_HasOpticalDiskDrive	int8
Census_TotalPhysicalRAM	float32
Census_ChassisTypeName	int8
Census_InternalPrimaryDiagonalDisplaySizeInInches	float16
Census_InternalPrimaryDisplayResolutionHorizontal	float16
Census_InternalPrimaryDisplayResolutionVertical	float16
Census_PowerPlatformRoleName	int8
Census_InternalBatteryType	int8
Census_InternalBatteryNumberOfCharges	float32
Census_OSVersion	int16
Census_OSArchitecture	int8
Census_OSBranch	int8
Census_OSBuildNumber	int16
Census_OSBuildRevision	int16
Census_OSEdition	int8
Census_OSInstallTypeName	int8
Census_OSUILocaleIdentifier	int16
Census_OSWUAutoUpdateOptionsName	int8
Census_GenuineStateName	int8
Census_ActivationChannel	int8
Census_FlightRing	int8
Census_FirmwareManufacturerIdentifier	float16

```
Census_FirmwareVersionIdentifier                    float32
Census_IsSecureBootEnabled                             int8
Census_IsTouchEnabled                                  int8
Census_IsPenCapable                                    int8
Census_IsAlwaysOnAlwaysConnectedCapable             float16
Wdft_IsGamer                                        float16
Wdft_RegionIdentifier                               float16
HasDetections                                          int8
dtype: object
Index(['ProductName', 'EngineVersion', 'AppVersion', 'AvSigVersion',
       'RtpStateBitfield', 'IsSxsPassiveMode', 'DefaultBrowsersIdentifier',
       'AVProductStatesIdentifier', 'AVProductsInstalled', 'AVProductsEnable
d',
       'HasTpm', 'CountryIdentifier', 'CityIdentifier',
       'OrganizationIdentifier', 'GeoNameIdentifier',
       'LocaleEnglishNameIdentifier', 'OsVer', 'OsBuild', 'OsSuite',
       'OsPlatformSubRelease', 'OsBuildLab', 'SkuEdition', 'IsProtected',
       'IeVerIdentifier', 'SmartScreen', 'Firewall', 'Census_MDC2FormFactor',
       'Census_OEMNameIdentifier', 'Census_OEMModelIdentifier',
       'Census_ProcessorCoreCount', 'Census_ProcessorManufacturerIdentifier',
       'Census_ProcessorModelIdentifier', 'Census_PrimaryDiskTotalCapacity',
       'Census_PrimaryDiskTypeName', 'Census_SystemVolumeTotalCapacity',
       'Census_HasOpticalDiskDrive', 'Census_TotalPhysicalRAM',
       'Census_ChassisTypeName',
       'Census_InternalPrimaryDiagonalDisplaySizeInInches',
       'Census_InternalPrimaryDisplayResolutionHorizontal',
       'Census_InternalPrimaryDisplayResolutionVertical',
       'Census_PowerPlatformRoleName', 'Census_InternalBatteryType',
       'Census_InternalBatteryNumberOfCharges', 'Census_OSVersion',
       'Census_OSArchitecture', 'Census_OSBranch', 'Census_OSBuildNumber',
       'Census_OSBuildRevision', 'Census_OSEdition',
       'Census_OSInstallTypeName', 'Census_OSUILocaleIdentifier',
       'Census_OSWUAutoUpdateOptionsName', 'Census_GenuineStateName',
       'Census_ActivationChannel', 'Census_FlightRing',
       'Census_FirmwareManufacturerIdentifier',
       'Census_FirmwareVersionIdentifier', 'Census_IsSecureBootEnabled',
       'Census_IsTouchEnabled', 'Census_IsPenCapable',
       'Census_IsAlwaysOnAlwaysConnectedCapable', 'Wdft_IsGamer',
       'Wdft_RegionIdentifier', 'HasDetections'],
      dtype='object')
```

In [49]:
```
# now we need to setup our test data set matching the train dataset EDA.
test_data=pd.read_csv(r'C:\Users\Dshac\OneDrive\Documents\CSPB 3022\Project\mi
crosoft-malware-prediction\test.csv', low_memory=False)
```

In [50]:
```
dtypes = {
        'MachineIdentifier':                                    'category',
        'ProductName':                                          'category',
        'EngineVersion':                                        'category',
        'AppVersion':                                           'category',
        'AvSigVersion':                                         'category',
        'IsBeta':                                               'int8',
        'RtpStateBitfield':                                     'float16',
        'IsSxsPassiveMode':                                     'int8',
        'DefaultBrowsersIdentifier':                            'float32',
        'AVProductStatesIdentifier':                            'float32',
        'AVProductsInstalled':                                  'float16',
        'AVProductsEnabled':                                    'float16',
        'HasTpm':                                               'int8',
        'CountryIdentifier':                                    'int16',
        'CityIdentifier':                                       'float32',
        'OrganizationIdentifier':                               'float16',
        'GeoNameIdentifier':                                    'float16',
        'LocaleEnglishNameIdentifier':                          'int16',
        'Platform':                                             'category',
        'Processor':                                            'category',
        'OsVer':                                                'category',
        'OsBuild':                                              'int16',
        'OsSuite':                                              'int16',
        'OsPlatformSubRelease':                                 'category',
        'OsBuildLab':                                           'category',
        'SkuEdition':                                           'category',
        'IsProtected':                                          'float16',
        'AutoSampleOptIn':                                      'int8',
        'PuaMode':                                              'category',
        'SMode':                                                'float16',
        'IeVerIdentifier':                                      'float16',
        'SmartScreen':                                          'category',
        'Firewall':                                             'float16',
        'UacLuaenable':                                         'float32',
        'UacLuaenable':                                         'float64', # w
as 'float32'
        'Census_MDC2FormFactor':                                'category',
        'Census_DeviceFamily':                                  'category',
        'Census_OEMNameIdentifier':                             'float32', # w
as 'float16'
        'Census_OEMModelIdentifier':                            'float32',
        'Census_ProcessorCoreCount':                            'float16',
        'Census_ProcessorManufacturerIdentifier':               'float16',
        'Census_ProcessorModelIdentifier':                      'float32', # w
as 'float16'
        'Census_ProcessorClass':                                'category',
        'Census_PrimaryDiskTotalCapacity':                      'float64', # w
as 'float32'
        'Census_PrimaryDiskTypeName':                           'category',
        'Census_SystemVolumeTotalCapacity':                     'float64', # w
as 'float32'
        'Census_HasOpticalDiskDrive':                           'int8',
        'Census_TotalPhysicalRAM':                              'float32',
        'Census_ChassisTypeName':                               'category',
        'Census_InternalPrimaryDiagonalDisplaySizeInInches':    'float32', # w
```

```
as 'float16'
        'Census_InternalPrimaryDisplayResolutionHorizontal':    'float32', # w
as 'float16'
        'Census_InternalPrimaryDisplayResolutionVertical':      'float32', # w
as 'float16'
        'Census_PowerPlatformRoleName':                         'category',
        'Census_InternalBatteryType':                           'category',
        'Census_InternalBatteryNumberOfCharges':                'float64', # w
as 'float32'
        'Census_OSVersion':                                     'category',
        'Census_OSArchitecture':                                'category',
        'Census_OSBranch':                                      'category',
        'Census_OSBuildNumber':                                 'int16',
        'Census_OSBuildRevision':                               'int32',
        'Census_OSEdition':                                     'category',
        'Census_OSSkuName':                                     'category',
        'Census_OSInstallTypeName':                             'category',
        'Census_OSInstallLanguageIdentifier':                   'float16',
        'Census_OSUILocaleIdentifier':                          'int16',
        'Census_OSWUAutoUpdateOptionsName':                     'category',
        'Census_IsPortableOperatingSystem':                     'int8',
        'Census_GenuineStateName':                              'category',
        'Census_ActivationChannel':                             'category',
        'Census_IsFlightingInternal':                           'float16',
        'Census_IsFlightsDisabled':                             'float16',
        'Census_FlightRing':                                    'category',
        'Census_ThresholdOptIn':                                'float16',
        'Census_FirmwareManufacturerIdentifier':                'float16',
        'Census_FirmwareVersionIdentifier':                     'float32',
        'Census_IsSecureBootEnabled':                           'int8',
        'Census_IsWIMBootEnabled':                              'float16',
        'Census_IsVirtualDevice':                               'float16',
        'Census_IsTouchEnabled':                                'int8',
        'Census_IsPenCapable':                                  'int8',
        'Census_IsAlwaysOnAlwaysConnectedCapable':              'float16',
        'Wdft_IsGamer':                                         'float16',
        'Wdft_RegionIdentifier':                                'float16',
        'HasDetections':                                        'int8'
        }

test_data.head()
```

Out[50]:

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion
0	0000010489e3af074adeac69c53e555e	win8defender	1.1.15400.5	4.18.1810.5	1.281.501.0
1	00000176ac758d54827acd545b6315a5	win8defender	1.1.15400.4	4.18.1809.2	1.279.301.0
2	0000019dcefc128c2d4387c1273dae1d	win8defender	1.1.15300.6	4.18.1809.2	1.277.230.0
3	0000055553dc51b1295785415f1a224d	win8defender	1.1.15400.5	4.18.1810.5	1.281.664.0
4	00000574cefffeca83ec8adf9285b2bf	win8defender	1.1.15400.4	4.18.1809.2	1.279.236.0

5 rows × 82 columns

In [100]:
```
# test_data=test_data.astype(dtypes)
test_data.dtypes
machining=test_data['MachineIdentifier']
```

In [52]:
```
# we need to have computation test data to test effectivity of fit.
temp_x=train_opt.values[:,:-1] #grab the train data that is not in detections
temp_y=train_opt['HasDetections'] #this is the target

train_x=temp_x[::2] #make train be every other element
train_y=temp_y[::2]
test_x=temp_x[1::2] #make test be every other element starting at 1
test_y=temp_y[1::2]
```

In [53]:
```
#Lets try a nearest neighbor approach...
```

In [54]:
```
# train_y=train_opt['HasDetections'] NO LONGER NEEDED
# train_x=train_opt.values[:,:-1]
print(train_y.shape)
print(train_x.shape)

# k = 1
# neigh = sklearn.neighbors.KNeighborsClassifier(n_neighbors= k)
# neigh.fit(carv_X_train, carv_y_train)

# knn_hat = neigh.predict(carv_X_test)
# knn_cm = sklearn.metrics.confusion_matrix(carv_y_test, knn_hat)
# print('KNN(', k,') confusion matrix is = \n', knn_cm)
# print('Success = ', knn_cm[1,1] / (knn_cm[1,0] + knn_cm[1,1]))
# print(sklearn.metrics.classification_report(carv_y_test, knn_hat))
```

```
(3833895,)
(3833895, 64)
```

In [55]:
```
k=1
%time neigh=sklearn.neighbors.KNeighborsClassifier(n_neighbors=k, algorithm='k
d_tree', p=1, n_jobs=-1)
%time neigh.fit(train_x[::10],train_y[::10])
#neigh.fit(train_x,train_y), not enough comp power to run this.
```

```
Wall time: 0 ns
Wall time: 3min 31s
```

Out[55]:
```
KNeighborsClassifier(algorithm='kd_tree', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=1, p=1,
                     weights='uniform')
```

In [ ]:
```
```

In [56]:
```
%time knn_hat = neigh.predict(test_x)
%time knn_hat_p = neigh.predict_proba(test_x)

knn_cm = sklearn.metrics.confusion_matrix(test_y, knn_hat)
print('KNN(', k,') confusion matrix is = \n', knn_cm)
print('Success = ', knn_cm[1,1] / (knn_cm[1,0] + knn_cm[1,1]))
print(sklearn.metrics.classification_report(test_y, knn_hat))
```

```
Wall time: 5min 34s
Wall time: 4min 3s
KNN( 1 ) confusion matrix is =
 [[ 972660  938471]
 [ 861933 1060830]]
Success =  0.5517216630442754
              precision    recall  f1-score   support

           0       0.53      0.51      0.52   1911131
           1       0.53      0.55      0.54   1922763

    accuracy                           0.53   3833894
   macro avg       0.53      0.53      0.53   3833894
weighted avg       0.53      0.53      0.53   3833894
```

In [57]:
```
#lets seee what a logistic regression can do....
print(knn_hat.shape)
print(test_y.shape)
```

```
(3833894,)
(3833894,)
```

In [58]:
```
# your code here
%time lreg=sklearn.linear_model.LogisticRegression(solver='lbfgs', max_iter=10
000, n_jobs=-1)
# print(mX_train.shape)
# print(my_train.shape)

# mX_train_shaped=np.reshape(mX_train,(60000,784))
# print(np.reshape(mX_train,(60000,784)))
# print(mX_train_shaped.shape)

%time lregm=lreg.fit(train_x,train_y)
```

```
Wall time: 0 ns
Wall time: 57.5 s
```

In [59]:
```
test_y.shape
```

Out[59]:
```
(3833894,)
```

In [60]:
```
# your code here
%time yhat_p=lregm.predict_proba(test_x)
%time yhat=lregm.predict(test_x)
```

```
Wall time: 1.79 s
Wall time: 1.6 s
```

In [61]:

```
print(test_y)
print(yhat)
scorer=pd.DataFrame(yhat)
print(scorer)
# new_hat=test_y.copy()
# print(yhat)
# print(new_hat)
# new_hat[:]=yhat
# print(new_hat)

print('For', test_x[0], 'we get probability', yhat_p[0], 'yhat is', yhat[0])
#print('Our logistic estimate is', logistic(lregm, mX_test_shaped))
# plt.plot(mX_test_shaped,yhat,'go');
# plt.plot(mX_test_shaped,yhat_p[:,1],'bo');
# #plt.plot(X, logistic(default_mod, X), 'ro')
# plt.axhline(0.5);
print('')
print('Logistic Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat, t
est_y))
print('')
print(sklearn.metrics.classification_report(test_y, yhat))
#print('Score is', lregm.score(test_y,scorer))
```

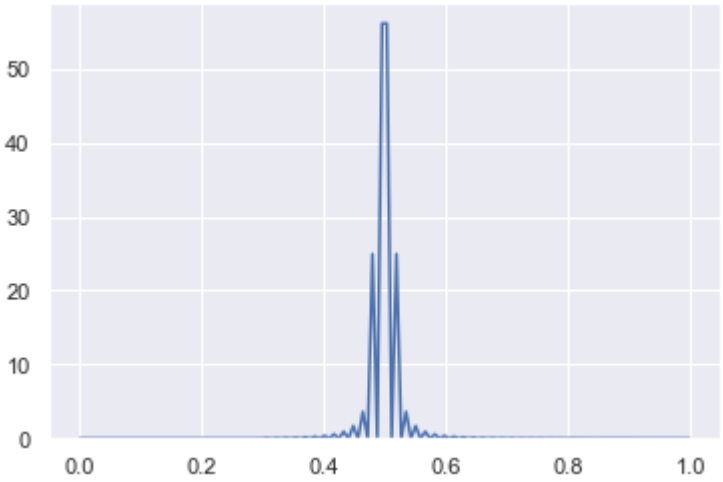
```
1          0
3          1
5          1
7          0
10         0
          ..
8921468    0
8921471    1
8921476    0
8921478    1
8921481    1
Name: HasDetections, Length: 3833894, dtype: int8
[0 1 1 ... 0 0 0]
         0
0        0
1        1
2        1
3        0
4        0
...     ..
3833889  0
3833890  0
3833891  0
3833892  0
3833893  0

[3833894 rows x 1 columns]
For [3.00000000e+00 5.00000000e+01 2.60000000e+01 6.24800000e+03
 7.00000000e+00 0.00000000e+00 0.00000000e+00 5.34470000e+04
 1.00000000e+00 1.00000000e+00 1.00000000e+00 9.30000000e+01
 1.48200000e+03 1.80000000e+01 1.19000000e+02 6.40000000e+01
 0.00000000e+00 1.71340000e+04 2.56000000e+02 4.00000000e+00
 2.57000000e+02 6.00000000e+00 1.00000000e+00 1.37000000e+02
 7.00000000e+00 1.00000000e+00 8.00000000e+00 2.66800000e+03
 9.16560000e+04 4.00000000e+00 5.00000000e+00 2.40400000e+03
 4.76940000e+05 0.00000000e+00 1.02385000e+05 0.00000000e+00
 4.09600000e+03 3.20000000e+01 1.38984375e+01 1.36600000e+03
 7.68000000e+02 3.00000000e+00 6.70000000e+01 1.00000000e+00
 3.30000000e+02 0.00000000e+00 9.00000000e+00 1.71340000e+04
 1.00000000e+00 1.60000000e+01 2.00000000e+00 3.10000000e+01
 4.00000000e+00 2.00000000e+00 2.00000000e+00 2.00000000e+00
 6.28000000e+02 5.78580000e+04 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 8.00000000e+00] we get probabil
ity [0.5000001 0.4999999] yhat is 0

Logistic Confusion Matrix:
 [[1425543 1404914]
 [ 485588  517849]]

              precision    recall  f1-score   support

           0       0.50      0.75      0.60   1911131
           1       0.52      0.27      0.35   1922763

    accuracy                           0.51   3833894
   macro avg       0.51      0.51      0.48   3833894
```

```
weighted avg       0.51      0.51      0.48   3833894
```

In [62]:
```
print(yhat_p.ravel())
sns.kdeplot(yhat_p.ravel())
```

```
[0.5000001  0.4999999  0.47918795 ... 0.49999981 0.50000009 0.49999991]
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16dcecadd88>
```



In [63]:
```
#Linear discriminant analysis should be superior with multinomial classificati
on....
```

In [64]:
```
lda = sklearn.discriminant_analysis.LinearDiscriminantAnalysis()
%time lda_mod = lda.fit(train_x, train_y)
```

```
Wall time: 38.6 s
```

In [65]:
```
%time yhat_p1=lda_mod.predict_proba(test_x)
%time yhat1=lda_mod.predict(test_x)

# print('LDA Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat, test
_y)

# print('Score is', 1-default_mod.score(test_x,test_y))
```

```
Wall time: 501 ms
Wall time: 457 ms
```

In [66]:
```
print(yhat1.shape)

print('For', test_x[0], 'we get probability', yhat_p1[0], 'yhat is', yhat1[0])
#print('Our logistic estimate is', logistic(lregm, mX_test_shaped))
# plt.plot(mX_test_shaped,yhat,'go');
# plt.plot(mX_test_shaped,yhat_p[:,1],'bo');
# #plt.plot(X, logistic(default_mod, X), 'ro')
# plt.axhline(0.5);
print('')
print('LDA Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat1, test_
y))
print('')
print(sklearn.metrics.classification_report(test_y, yhat1))
#print('Score is', lregm.score(test_y,yhat))
```

```
(3833894,)
For [3.00000000e+00 5.00000000e+01 2.60000000e+01 6.24800000e+03
 7.00000000e+00 0.00000000e+00 0.00000000e+00 5.34470000e+04
 1.00000000e+00 1.00000000e+00 1.00000000e+00 9.30000000e+01
 1.48200000e+03 1.80000000e+01 1.19000000e+02 6.40000000e+01
 0.00000000e+00 1.71340000e+04 2.56000000e+02 4.00000000e+00
 2.57000000e+02 6.00000000e+00 1.00000000e+00 1.37000000e+02
 7.00000000e+00 1.00000000e+00 8.00000000e+00 2.66800000e+03
 9.16560000e+04 4.00000000e+00 5.00000000e+00 2.40400000e+03
 4.76940000e+05 0.00000000e+00 1.02385000e+05 0.00000000e+00
 4.09600000e+03 3.20000000e+01 1.38984375e+01 1.36600000e+03
 7.68000000e+02 3.00000000e+00 6.70000000e+01 1.00000000e+00
 3.30000000e+02 0.00000000e+00 9.00000000e+00 1.71340000e+04
 1.00000000e+00 1.60000000e+01 2.00000000e+00 3.10000000e+01
 4.00000000e+00 2.00000000e+00 2.00000000e+00 2.00000000e+00
 6.28000000e+02 5.78580000e+04 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 8.00000000e+00] we get probabil
ity [0.42109436 0.57890564] yhat is 1

LDA Confusion Matrix:
 [[1065172  674539]
 [ 845959 1248224]]

              precision    recall  f1-score   support

           0       0.61      0.56      0.58   1911131
           1       0.60      0.65      0.62   1922763

    accuracy                           0.60   3833894
   macro avg       0.60      0.60      0.60   3833894
weighted avg       0.60      0.60      0.60   3833894
```

In [67]:
```
qda = sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis()
%time qda_mod = qda.fit(train_x, train_y)
```

```
Wall time: 14.3 s
```

In [68]:
```
%time yhat_p=qda_mod.predict_proba(test_x)
%time yhat=qda_mod.predict(test_x)

# print('LDA Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat, test
_y)

# print('Score is', 1-default_mod.score(test_x,test_y))
```

```
Wall time: 8.19 s
Wall time: 8.1 s
```

In [69]:
```
print(yhat.shape)

print('For', test_x[0], 'we get probability', yhat_p[0], 'yhat is', yhat[0])
#print('Our logistic estimate is', logistic(lregm, mX_test_shaped))
# plt.plot(mX_test_shaped,yhat,'go');
# plt.plot(mX_test_shaped,yhat_p[:,1],'bo');
# #plt.plot(X, logistic(default_mod, X), 'ro')
# plt.axhline(0.5);
print('')
print('LDA Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat, test_y
))
print('')
print(sklearn.metrics.classification_report(test_y, yhat))
#print('Score is', lregm.score(test_y,yhat))
```

```
(3833894,)
For [3.00000000e+00 5.00000000e+01 2.60000000e+01 6.24800000e+03
 7.00000000e+00 0.00000000e+00 0.00000000e+00 5.34470000e+04
 1.00000000e+00 1.00000000e+00 1.00000000e+00 9.30000000e+01
 1.48200000e+03 1.80000000e+01 1.19000000e+02 6.40000000e+01
 0.00000000e+00 1.71340000e+04 2.56000000e+02 4.00000000e+00
 2.57000000e+02 6.00000000e+00 1.00000000e+00 1.37000000e+02
 7.00000000e+00 1.00000000e+00 8.00000000e+00 2.66800000e+03
 9.16560000e+04 4.00000000e+00 5.00000000e+00 2.40400000e+03
 4.76940000e+05 0.00000000e+00 1.02385000e+05 0.00000000e+00
 4.09600000e+03 3.20000000e+01 1.38984375e+01 1.36600000e+03
 7.68000000e+02 3.00000000e+00 6.70000000e+01 1.00000000e+00
 3.30000000e+02 0.00000000e+00 9.00000000e+00 1.71340000e+04
 1.00000000e+00 1.60000000e+01 2.00000000e+00 3.10000000e+01
 4.00000000e+00 2.00000000e+00 2.00000000e+00 2.00000000e+00
 6.28000000e+02 5.78580000e+04 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 8.00000000e+00] we get probabil
ity [2.07455716e-07 9.99999793e-01] yhat is 1

LDA Confusion Matrix:
 [[ 160885   84038]
 [1750246 1838725]]

              precision    recall  f1-score   support

           0       0.66      0.08      0.15   1911131
           1       0.51      0.96      0.67   1922763

    accuracy                           0.52   3833894
   macro avg       0.58      0.52      0.41   3833894
weighted avg       0.58      0.52      0.41   3833894
```

In [70]:
```
#Our KNN test resulted in an accuracy score of: 0.55, but could not be run on
 the full data set
#Our Logistic test resulted in an accuracy score of: 0.5
#Our LDA test resulted in an accuracy score of: 0.6
#Because these accuracy scores are still too low, lets see if there is any mor
e feature optimization we can do.....
#Although not introduced in class, lets see if we can figure out how to run a
 Recursive feature elimination analysis...
```

In [71]:
```
data_final_vars=train_data.columns.values.tolist()
print(data_final_vars)
y=['HasDetections']
X=[i for i in data_final_vars if i not in y]


from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression


logreg=LogisticRegression()

rfe= RFE(logreg, 20)

%time rfe=rfe.fit(train_x, train_y)
```

```
['ProductName', 'EngineVersion', 'AppVersion', 'AvSigVersion', 'RtpStateBitfi
eld', 'IsSxsPassiveMode', 'DefaultBrowsersIdentifier', 'AVProductStatesIdenti
fier', 'AVProductsInstalled', 'AVProductsEnabled', 'HasTpm', 'CountryIdentifi
er', 'CityIdentifier', 'OrganizationIdentifier', 'GeoNameIdentifier', 'Locale
EnglishNameIdentifier', 'OsVer', 'OsBuild', 'OsSuite', 'OsPlatformSubReleas
e', 'OsBuildLab', 'SkuEdition', 'IsProtected', 'IeVerIdentifier', 'SmartScree
n', 'Firewall', 'Census_MDC2FormFactor', 'Census_OEMNameIdentifier', 'Census_
OEMModelIdentifier', 'Census_ProcessorCoreCount', 'Census_ProcessorManufactur
erIdentifier', 'Census_ProcessorModelIdentifier', 'Census_PrimaryDiskTotalCap
acity', 'Census_PrimaryDiskTypeName', 'Census_SystemVolumeTotalCapacity', 'Ce
nsus_HasOpticalDiskDrive', 'Census_TotalPhysicalRAM', 'Census_ChassisTypeNam
e', 'Census_InternalPrimaryDiagonalDisplaySizeInInches', 'Census_InternalPrim
aryDisplayResolutionHorizontal', 'Census_InternalPrimaryDisplayResolutionVert
ical', 'Census_PowerPlatformRoleName', 'Census_InternalBatteryType', 'Census_
InternalBatteryNumberOfCharges', 'Census_OSVersion', 'Census_OSArchitecture',
'Census_OSBranch', 'Census_OSBuildNumber', 'Census_OSBuildRevision', 'Census_
OSEdition', 'Census_OSInstallTypeName', 'Census_OSUILocaleIdentifier', 'Censu
s_OSWUAutoUpdateOptionsName', 'Census_GenuineStateName', 'Census_ActivationCh
annel', 'Census_FlightRing', 'Census_FirmwareManufacturerIdentifier', 'Census
_FirmwareVersionIdentifier', 'Census_IsSecureBootEnabled', 'Census_IsTouchEna
bled', 'Census_IsPenCapable', 'Census_IsAlwaysOnAlwaysConnectedCapable', 'Wdf
t_IsGamer', 'Wdft_RegionIdentifier', 'HasDetections']
Wall time: 5min 39s
```

In [72]:
```
print(rfe.support_)
print(rfe.ranking_)
ranks=rfe.ranking_
```

```
[False False False  True False False  True  True False False False False
  True False False False False  True  True False False False False False
 False False False  True  True False False  True  True False  True False
  True False False  True  True False False  True  True False False  True
  True False False False False False False False  True  True False False
 False False False False]
[28 14  3  1 44 39  1  1 12 33 36  7  1 19  2  6 34  1  1 40 11 27 35 15
  4 37 26  1  1 13 29  1  1 25  1 38  1  9 20  1  1 17  8  1  1 21 31  1
  1 16 18  5 23 42 43 22  1  1 45 32 41 30 24 10]
```

In [73]:
```
cols=[]
for i in range(len(ranks)):
    if (ranks[i]==1):
        cols.append(data_final_vars[i])
print(cols)
```

```
['AvSigVersion', 'DefaultBrowsersIdentifier', 'AVProductStatesIdentifier', 'C
ityIdentifier', 'OsBuild', 'OsSuite', 'Census_OEMNameIdentifier', 'Census_OEM
ModelIdentifier', 'Census_ProcessorModelIdentifier', 'Census_PrimaryDiskTotal
Capacity', 'Census_SystemVolumeTotalCapacity', 'Census_TotalPhysicalRAM', 'Ce
nsus_InternalPrimaryDisplayResolutionHorizontal', 'Census_InternalPrimaryDisp
layResolutionVertical', 'Census_InternalBatteryNumberOfCharges', 'Census_OSVe
rsion', 'Census_OSBuildNumber', 'Census_OSBuildRevision', 'Census_FirmwareMan
ufacturerIdentifier', 'Census_FirmwareVersionIdentifier']
```

In [74]:
```
X=train_data[cols]
y=train_data['HasDetections']

import statsmodels.api as sm
logit_model=sm.Logit(y,X)

result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.680620
         Iterations 4
                                          Results: Logit
================================================================================
=====================
Model:                          Logit                       Pseudo R-squared:
0.018
Dependent Variable:             HasDetections               AIC:
10437698.7900
Date:                           2020-08-10 13:12            BIC:
10437712.6425
No. Observations:               7667789                     Log-Likelihood:
-5.2188e+06
Df Model:                       0                           LL-Null:
-5.3149e+06
Df Residuals:                   7667788                     LLR p-value:
nan
Converged:                      1.0000                      Scale:
1.0000
No. Iterations:                 4.0000
--------------------------------------------------------------------------------
---------------------
                                                    Coef.  Std.Err.     z
P>|z|   [0.025  0.975]
--------------------------------------------------------------------------------
---------------------
AvSigVersion                                         0.0001   0.0000   94.4815
0.0000  0.0001  0.0001
DefaultBrowsersIdentifier                            0.0001   0.0000   31.9782
0.0000  0.0001  0.0001
AVProductStatesIdentifier                            0.0000   0.0000  328.8683
0.0000  0.0000  0.0000
CityIdentifier                                      -0.0000   0.0000  -12.7725
0.0000 -0.0000 -0.0000
OsBuild                                             -0.0000   0.0000  -12.0924
0.0000 -0.0000 -0.0000
OsSuite                                             -0.0001   0.0000  -25.1597
0.0000 -0.0001 -0.0001
Census_OEMNameIdentifier                            -0.0000   0.0000  -30.4047
0.0000 -0.0000 -0.0000
Census_OEMModelIdentifier                           -0.0000   0.0000  -31.4513
0.0000 -0.0000 -0.0000
Census_ProcessorModelIdentifier                      0.0000   0.0000   27.1579
0.0000  0.0000  0.0000
Census_PrimaryDiskTotalCapacity                      0.0000   0.0000    0.5754
0.5650 -0.0000  0.0000
Census_SystemVolumeTotalCapacity                     0.0000   0.0000   25.8192
0.0000  0.0000  0.0000
Census_TotalPhysicalRAM                              0.0000   0.0000  115.8030
0.0000  0.0000  0.0000
Census_InternalPrimaryDisplayResolutionHorizontal    0.0005   0.0000   97.7643
0.0000  0.0005  0.0005
Census_InternalPrimaryDisplayResolutionVertical     -0.0010   0.0000 -116.0479
0.0000 -0.0011 -0.0010
Census_InternalBatteryNumberOfCharges                0.0000   0.0000   13.2487
0.0000  0.0000  0.0000
```

```
Census_OSVersion                                  0.0043   0.0000  179.2858
0.0000  0.0042  0.0043
Census_OSBuildNumber                             -0.0001   0.0000  -97.9788
0.0000 -0.0002 -0.0001
Census_OSBuildRevision                            0.0000   0.0000   19.4770
0.0000  0.0000  0.0000
Census_FirmwareManufacturerIdentifier            -0.0001   0.0000  -36.4686
0.0000 -0.0002 -0.0001
Census_FirmwareVersionIdentifier                 -0.0000   0.0000   -4.8008
0.0000 -0.0000 -0.0000
================================================================================
======================
```

In [75]:
```
#p values for Census_PrimaryDiskTotalCapacity and Census_FirmwareVersionIdenti
fier are much larger than the others,
#lets drop them from our data....
#cols=cols.remove('Census_PrimaryDiskTotalCapacity')
cols= ['AvSigVersion', 'DefaultBrowsersIdentifier', 'AVProductStatesIdentifie
r', 'CityIdentifier', 'LocaleEnglishNameIdentifier', 'OsBuild', 'OsSuite', 'Ce
nsus_OEMNameIdentifier', 'Census_OEMModelIdentifier', 'Census_ProcessorModelId
entifier', 'Census_SystemVolumeTotalCapacity', 'Census_TotalPhysicalRAM', 'Cen
sus_InternalPrimaryDisplayResolutionHorizontal', 'Census_InternalBatteryNumber
OfCharges', 'Census_OSVersion', 'Census_OSBuildNumber', 'Census_OSBuildRevisio
n', 'Census_FirmwareManufacturerIdentifier','Census_FirmwareVersionIdentifier'
]
print(cols)
X=train_data[cols]
y=train_data['HasDetections']


temp_x=X.values[:,:-1] #grab te train data taht is not in detections
temp_y=y #this is the target

train_x=temp_x[::2] #make train be every other element
train_y=temp_y[::2]
test_x=temp_x[1::2] #make test be every other element starting at 1
test_y=temp_y[1::2]

logit_model=sm.Logit(train_y,train_x)

result=logit_model.fit()
print(result.summary2())
```

['AvSigVersion', 'DefaultBrowsersIdentifier', 'AVProductStatesIdentifier', 'CityIdentifier', 'LocaleEnglishNameIdentifier', 'OsBuild', 'OsSuite', 'Census_OEMNameIdentifier', 'Census_OEMModelIdentifier', 'Census_ProcessorModelIdentifier', 'Census_SystemVolumeTotalCapacity', 'Census_TotalPhysicalRAM', 'Census_InternalPrimaryDisplayResolutionHorizontal', 'Census_InternalBatteryNumberOfCharges', 'Census_OSVersion', 'Census_OSBuildNumber', 'Census_OSBuildRevision', 'Census_FirmwareManufacturerIdentifier', 'Census_FirmwareVersionIdentifier']
Optimization terminated successfully.
         Current function value: 0.681213
         Iterations 4
                         Results: Logit
==
Model: Logit   Pseudo R-squared: 0.017
Dependent Variable: HasDetections   AIC: 5223401.6821
Date: 2020-08-10 13:12 BIC: 5223414.8415
No. Observations: 3833895   Log-Likelihood: -2.6117e+06
Df Model: 0   LL-Null: -2.6574e+06
Df Residuals: 3833894   LLR p-value: nan
Converged: 1.0000   Scale: 1.0000
No. Iterations: 4.0000

        Coef.   Std.Err.   z   P>|z| [0.025   0.975]
----
x1   0.0001   0.0000   63.0264   0.0000   0.0001   0.0001
x2   0.0001   0.0000   20.7771   0.0000   0.0001   0.0001
x3   0.0000   0.0000   232.4603   0.0000   0.0000   0.0000
x4   -0.0000   0.0000   -11.2574   0.0000   -0.0000   -0.0000
x5   0.0008   0.0000   55.2747   0.0000   0.0008   0.0009
x6   -0.0000   0.0000   -8.2511   0.0000   -0.0000   -0.0000
x7   -0.0001   0.0000   -14.3408   0.0000   -0.0001   -0.0001
x8   -0.0000   0.0000   -28.4948   0.0000   -0.0000   -0.0000
x9   -0.0000   0.0000   -29.1024   0.0000   -0.0000   -0.0000
x10   0.0000   0.0000   19.1827   0.0000   0.0000   0.0000
x11   0.0000   0.0000   18.5292   0.0000   0.0000   0.0000
x12   0.0000   0.0000   85.6306   0.0000   0.0000   0.0000
x13   -0.0000   0.0000   -5.6422   0.0000   -0.0000   -0.0000
x14   -0.0000   0.0000   -4.4572   0.0000   -0.0000   -0.0000
x15   0.0046   0.0000   135.2295   0.0000   0.0045   0.0046
x16   -0.0002   0.0000   -75.9332   0.0000   -0.0002   -0.0002
x17   0.0000   0.0000   15.1923   0.0000   0.0000   0.0000
x18   -0.0002   0.0000   -27.1079   0.0000   -0.0002   -0.0001
==

In [76]:
```
logreg = LogisticRegression()
logreg.fit(train_x, train_y)

test_x.shape
train_x.shape


%time yhat_p=logreg.predict_proba(test_x)
%time yhat=logreg.predict(test_x)

print(yhat.shape)

print('For', test_x[0], 'we get probability', yhat_p[0], 'yhat is', yhat[0])
#print('Our logistic estimate is', logistic(lregm, mX_test_shaped))
# plt.plot(mX_test_shaped,yhat,'go');
# plt.plot(mX_test_shaped,yhat_p[:,1],'bo');
# #plt.plot(X, logistic(default_mod, X), 'ro')
# plt.axhline(0.5);
print('')
print('Logistic Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat, t
est_y))
print('')
print(sklearn.metrics.classification_report(test_y, yhat))
# print('Score is', lregm.score(yhat,test_y))
```

```
Wall time: 453 ms
Wall time: 413 ms
(3833894,)
For [6.24800e+03 0.00000e+00 5.34470e+04 1.48200e+03 6.40000e+01 1.71340e+04
 2.56000e+02 2.66800e+03 9.16560e+04 2.40400e+03 1.02385e+05 4.09600e+03
 1.36600e+03 1.00000e+00 3.30000e+02 1.71340e+04 1.00000e+00 6.28000e+02] we
get probability [0.5 0.5] yhat is 1

Logistic Confusion Matrix:
 [[ 298349  267778]
 [1612782 1654985]]

              precision    recall  f1-score   support

           0       0.53      0.16      0.24   1911131
           1       0.51      0.86      0.64   1922763

    accuracy                           0.51   3833894
   macro avg       0.52      0.51      0.44   3833894
weighted avg       0.52      0.51      0.44   3833894
```

In [77]:
```
#this is still not much better.... lets try the KNN again but with the two col
umns data we found weeded out....
```

In [78]:
```
print(test_x.shape)
print(test_y.shape)
```

```
(3833894, 18)
(3833894,)
```

In [79]:
```
dtypes = {
        'ProductName':                                          'int8',
        'EngineVersion':                                        'int8',
        'AppVersion':                                           'int8',
        'AvSigVersion':                                         'int16',
        'RtpStateBitfield':                                     'float16',
        'IsSxsPassiveMode':                                     'int8',
        'DefaultBrowsersIdentifier':                            'float16',
        'AVProductStatesIdentifier':                            'float32',
        'AVProductsInstalled':                                  'float16',
        'AVProductsEnabled':                                    'float16',
        'HasTpm':                                               'int8',
        'CountryIdentifier':                                    'int16',
        'CityIdentifier':                                       'float32',
        'OrganizationIdentifier':                               'float16',
        'GeoNameIdentifier':                                    'float16',
        'LocaleEnglishNameIdentifier':                          'int16',
        'OsVer':                                                'int8',
        'OsBuild':                                              'int16',
        'OsSuite':                                              'int16',
        'OsPlatformSubRelease':                                 'int8',
        'OsBuildLab':                                           'int16',
        'SkuEdition':                                           'int8',
        'IsProtected':                                          'float16',
        'IeVerIdentifier':                                      'float16',
        'SmartScreen':                                          'int8',
        'Firewall':                                             'float16',
        'Census_MDC2FormFactor':                                'int8',
        'Census_OEMNameIdentifier':                             'float16', # w
as 'float16'
        'Census_OEMModelIdentifier':                            'float32',
        'Census_ProcessorCoreCount':                            'float16',
        'Census_ProcessorManufacturerIdentifier':               'float16',
        'Census_ProcessorModelIdentifier':                      'float16', # w
as 'float16'
        'Census_PrimaryDiskTotalCapacity':                      'float32', # w
as 'float32'
        'Census_PrimaryDiskTypeName':                           'int8',
        'Census_SystemVolumeTotalCapacity':                     'float32', # w
as 'float32'
        'Census_HasOpticalDiskDrive':                           'int8',
        'Census_TotalPhysicalRAM':                              'float32',
        'Census_ChassisTypeName':                               'int8',
        'Census_InternalPrimaryDiagonalDisplaySizeInInches':    'float16', # w
as 'float16'
        'Census_InternalPrimaryDisplayResolutionHorizontal':    'float16', # w
as 'float16'
        'Census_InternalPrimaryDisplayResolutionVertical':      'float16', # w
as 'float16'
        'Census_PowerPlatformRoleName':                         'int8',
        'Census_InternalBatteryType':                           'int8',
        'Census_InternalBatteryNumberOfCharges':                'float32', # w
as 'float32'
        'Census_OSVersion':                                     'int16',
        'Census_OSArchitecture':                                'int8',
        'Census_OSBranch':                                      'int8',
```

```
        'Census_OSBuildNumber':                                 'int16',
        'Census_OSBuildRevision':                               'int16',
        'Census_OSEdition':                                     'int8',
        'Census_OSInstallTypeName':                             'int8',
        'Census_OSUILocaleIdentifier':                          'int16',
        'Census_OSWUAutoUpdateOptionsName':                     'int8',
        'Census_GenuineStateName':                              'int8',
        'Census_ActivationChannel':                             'int8',
        'Census_FlightRing':                                    'int8',
        'Census_FirmwareManufacturerIdentifier':                'float16',
        'Census_FirmwareVersionIdentifier':                     'float32',
        'Census_IsSecureBootEnabled':                           'int8',
        'Census_IsTouchEnabled':                                'int8',
        'Census_IsPenCapable':                                  'int8',
        'Census_IsAlwaysOnAlwaysConnectedCapable':              'float16',
        'Wdft_IsGamer':                                         'float16',
        'Wdft_RegionIdentifier':                                'float16',
        }
```

In [80]:
```
#now we can attempt to run our test_data through our LDA as it is the current
 best choice.
test_final=test_data.copy(deep=False)
additional_drop=[]

for col in test_final.columns:
    col_type = test_final[col].dtype

    if col_type == object:
        try:
            test_final[col].astype('int8')
        except:
            if col not in drop_features:
                additional_drop.append(col)

print(additional_drop)



test_final.drop(drop_features, axis=1, inplace=True)

test_final.drop(additional_drop, axis=1, inplace=True)
test_final = reduce_mem_usage(test_final)
# #test_final.astype(dtypes)
test_final.dtypes
```

```
['MachineIdentifier', 'ProductName', 'EngineVersion', 'AppVersion', 'AvSigVer
sion', 'OsVer', 'OsPlatformSubRelease', 'OsBuildLab', 'SkuEdition', 'SmartScr
een', 'Census_MDC2FormFactor', 'Census_PrimaryDiskTypeName', 'Census_ChassisT
ypeName', 'Census_PowerPlatformRoleName', 'Census_InternalBatteryType', 'Cens
us_OSVersion', 'Census_OSArchitecture', 'Census_OSBranch', 'Census_OSEditio
n', 'Census_OSInstallTypeName', 'Census_OSWUAutoUpdateOptionsName', 'Census_G
enuineStateName', 'Census_ActivationChannel', 'Census_FlightRing']
Memory usage of dataframe is 2456.54 MB
Memory usage after optimization is: 704.01 MB
Decreased by 71.3%
```

Out[80]:

```
RtpStateBitfield                                  float16
IsSxsPassiveMode                                    int8
DefaultBrowsersIdentifier                         float16
AVProductStatesIdentifier                         float32
AVProductsInstalled                               float16
AVProductsEnabled                                 float16
HasTpm                                              int8
CountryIdentifier                                 int16
CityIdentifier                                    float32
OrganizationIdentifier                            float16
GeoNameIdentifier                                 float16
LocaleEnglishNameIdentifier                       int16
OsBuild                                           int16
OsSuite                                           int16
IsProtected                                       float16
IeVerIdentifier                                   float16
Firewall                                          float16
Census_OEMNameIdentifier                          float16
Census_OEMModelIdentifier                         float32
Census_ProcessorCoreCount                         float16
Census_ProcessorManufacturerIdentifier            float16
Census_ProcessorModelIdentifier                   float16
Census_PrimaryDiskTotalCapacity                   float32
Census_SystemVolumeTotalCapacity                  float32
Census_HasOpticalDiskDrive                          int8
Census_TotalPhysicalRAM                           float32
Census_InternalPrimaryDiagonalDisplaySizeInInches float16
Census_InternalPrimaryDisplayResolutionHorizontal float16
Census_InternalPrimaryDisplayResolutionVertical   float16
Census_InternalBatteryNumberOfCharges             float32
Census_OSBuildNumber                              int16
Census_OSBuildRevision                            int32
Census_OSUILocaleIdentifier                       int16
Census_FirmwareManufacturerIdentifier             float16
Census_FirmwareVersionIdentifier                  float32
Census_IsSecureBootEnabled                          int8
Census_IsTouchEnabled                               int8
Census_IsPenCapable                                 int8
Census_IsAlwaysOnAlwaysConnectedCapable           float16
Wdft_IsGamer                                      float16
Wdft_RegionIdentifier                             float16
dtype: object
```

In [81]:

```
test_final.replace({'OrganizationIdentifier': {np.nan: 0}}, inplace=True)
```

In [82]:
```
trans_dict = {
    'off': 'Off', '&#x02;': '2', '&#x01;': '1', 'on': 'On', 'requireadmin': 'RequireAdmin', 'OFF': 'Off',
    'Promt': 'Prompt', 'requireAdmin': 'RequireAdmin', 'prompt': 'Prompt', 'warn': 'Warn',
    '00000000': '0', '&#x03;': '3', np.nan: 'NoExist'
}
test_final.replace({'SmartScreen': trans_dict}, inplace=True)

# trans_dict = {
#     '□···': 'unknown', 'unkn': 'unknown', np.nan: 'unknown'
# }
# train_data.replace({'Census_InternalBatteryType': trans_dict}, inplace=True)
```

In [83]:
```
test_final=np.nan_to_num(test_final)
```

In [ ]:
```

```

In [84]:
```
#newTrainData=train_data.drop(drop_features, axis=1, inplace=True)
additional_drop=['ProductName', 'EngineVersion', 'AppVersion', 'AvSigVersion', 'OsVer', 'OsPlatformSubRelease', 'OsBuildLab', 'SkuEdition', 'SmartScreen', 'Census_MDC2FormFactor', 'Census_PrimaryDiskTypeName', 'Census_ChassisTypeName', 'Census_PowerPlatformRoleName', 'Census_InternalBatteryType', 'Census_OSVersion', 'Census_OSArchitecture', 'Census_OSBranch', 'Census_OSEdition', 'Census_OSInstallTypeName', 'Census_OSWUAutoUpdateOptionsName', 'Census_GenuineStateName', 'Census_ActivationChannel', 'Census_FlightRing']
newTrainData=train_data.drop(additional_drop, axis=1, inplace=True)
```

In [85]:
```
print(train_data.shape)
train_data.dtypes
```

```
(7667789, 42)
```

Out[85]:

RtpStateBitfield	float16
IsSxsPassiveMode	int8
DefaultBrowsersIdentifier	float16
AVProductStatesIdentifier	float32
AVProductsInstalled	float16
AVProductsEnabled	float16
HasTpm	int8
CountryIdentifier	int16
CityIdentifier	float32
OrganizationIdentifier	float16
GeoNameIdentifier	float16
LocaleEnglishNameIdentifier	int16
OsBuild	int16
OsSuite	int16
IsProtected	float16
IeVerIdentifier	float16
Firewall	float16
Census_OEMNameIdentifier	float16
Census_OEMModelIdentifier	float32
Census_ProcessorCoreCount	float16
Census_ProcessorManufacturerIdentifier	float16
Census_ProcessorModelIdentifier	float16
Census_PrimaryDiskTotalCapacity	float32
Census_SystemVolumeTotalCapacity	float32
Census_HasOpticalDiskDrive	int8
Census_TotalPhysicalRAM	float32
Census_InternalPrimaryDiagonalDisplaySizeInInches	float16
Census_InternalPrimaryDisplayResolutionHorizontal	float16
Census_InternalPrimaryDisplayResolutionVertical	float16
Census_InternalBatteryNumberOfCharges	float32
Census_OSBuildNumber	int16
Census_OSBuildRevision	int16
Census_OSUILocaleIdentifier	int16
Census_FirmwareManufacturerIdentifier	float16
Census_FirmwareVersionIdentifier	float32
Census_IsSecureBootEnabled	int8
Census_IsTouchEnabled	int8
Census_IsPenCapable	int8
Census_IsAlwaysOnAlwaysConnectedCapable	float16
Wdft_IsGamer	float16
Wdft_RegionIdentifier	float16
HasDetections	int8

dtype: object

In [86]:
```
target_train.shape
```

Out[86]:
```
(7667789,)
```

In [87]:
```
test_final.shape
```

Out[87]:
```
(7853253, 41)
```

In [92]:
```
train_data.drop('HasDetections', axis=1, inplace=True)
train_data.dtypes
```

Out[92]:
```
RtpStateBitfield                                  float16
IsSxsPassiveMode                                    int8
DefaultBrowsersIdentifier                         float16
AVProductStatesIdentifier                         float32
AVProductsInstalled                               float16
AVProductsEnabled                                 float16
HasTpm                                              int8
CountryIdentifier                                 int16
CityIdentifier                                    float32
OrganizationIdentifier                            float16
GeoNameIdentifier                                 float16
LocaleEnglishNameIdentifier                       int16
OsBuild                                           int16
OsSuite                                           int16
IsProtected                                       float16
IeVerIdentifier                                   float16
Firewall                                          float16
Census_OEMNameIdentifier                          float16
Census_OEMModelIdentifier                         float32
Census_ProcessorCoreCount                         float16
Census_ProcessorManufacturerIdentifier            float16
Census_ProcessorModelIdentifier                   float16
Census_PrimaryDiskTotalCapacity                   float32
Census_SystemVolumeTotalCapacity                  float32
Census_HasOpticalDiskDrive                          int8
Census_TotalPhysicalRAM                           float32
Census_InternalPrimaryDiagonalDisplaySizeInInches float16
Census_InternalPrimaryDisplayResolutionHorizontal float16
Census_InternalPrimaryDisplayResolutionVertical   float16
Census_InternalBatteryNumberOfCharges             float32
Census_OSBuildNumber                              int16
Census_OSBuildRevision                            int16
Census_OSUILocaleIdentifier                       int16
Census_FirmwareManufacturerIdentifier             float16
Census_FirmwareVersionIdentifier                  float32
Census_IsSecureBootEnabled                          int8
Census_IsTouchEnabled                               int8
Census_IsPenCapable                                 int8
Census_IsAlwaysOnAlwaysConnectedCapable           float16
Wdft_IsGamer                                      float16
Wdft_RegionIdentifier                             float16
dtype: object
```

In [93]:
```
lda = sklearn.discriminant_analysis.LinearDiscriminantAnalysis()
%time lda_mod = lda.fit(train_data, target_train)
```

```
Wall time: 48 s
```

In [94]:
```
#now we can attempt to run our test_data through our LDA as it is the current
 best choice.

%time yhat_p1=lda_mod.predict_proba(test_final)
%time yhat1=lda_mod.predict(test_final)
# print('LDA Confusion Matrix:\n', sklearn.metrics.confusion_matrix(yhat, test
_y)
# print('Score is', 1-default_mod.score(test_x,test_y))
print('For', test_x[0], 'we get probability', yhat_p1[0], 'yhat is', yhat1[0])
```

```
Wall time: 629 ms
Wall time: 548 ms
For [6.24800e+03 0.00000e+00 5.34470e+04 1.48200e+03 6.40000e+01 1.71340e+04
 2.56000e+02 2.66800e+03 9.16560e+04 2.40400e+03 1.02385e+05 4.09600e+03
 1.36600e+03 1.00000e+00 3.30000e+02 1.71340e+04 1.00000e+00 6.28000e+02] we
get probability [0.44167218 0.55832782] yhat is 1
```

In [97]:
```
print(yhat1)
print(len(yhat1))
```

```
[1 1 0 ... 1 1 1]
7853253
```

In [103]:
```
import csv


with open('Shackelford.csv','w',newline='') as file:
    writer=csv.writer(file)
    writer.writerow(['MachineIdentifier','HasDetections'])
    for i in range(len(yhat1)):
        writer.writerow([machining[i],yhat1[i]])
```

In [91]:
```
halt=
k=1
%time neigh=sklearn.neighbors.KNeighborsClassifier(n_neighbors=k, algorithm='k
d_tree', p=1, n_jobs=-1)
%time neigh.fit(test_x[::2],test_y[::2])
#neigh.fit(train_x,train_y), not enough comp power to run this.
```

```
  File "<ipython-input-91-53d102b686d7>", line 1
    halt=
         ^
SyntaxError: invalid syntax
```

In [ ]: