In [81]:

```
#!/usr/bin/env python
# coding: utf-8
```

In [82]:

```
# Import dataset (csv.file)
import pandas as pd
df = pd.read_csv('online_shoppers_intention.csv')
```

In [83]:

```
df.head()
```

Out[83]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated
0	0	0.0	0.0	0.0	1
1	0	0.0	0.0	0.0	2
2	0	0.0	0.0	0.0	1
3	0	0.0	0.0	0.0	2
4	0	0.0	0.0	0.0	10

In [84]:

```
# Describe column dimension
numberofcolumns = len(df.columns)
print(numberofcolumns)
```

```
18
```

In [85]:

```
# Describe row dimension
numberofrows = len(df.index)
print(numberofrows)
```

```
12330
```

In [86]:

```
# Print column names
print(df.columns)
```

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

In [87]:

```
# Check data type
type(df)
```

Out[87]:

```
pandas.core.frame.DataFrame
```

In [88]:

```
import numpy as np
```

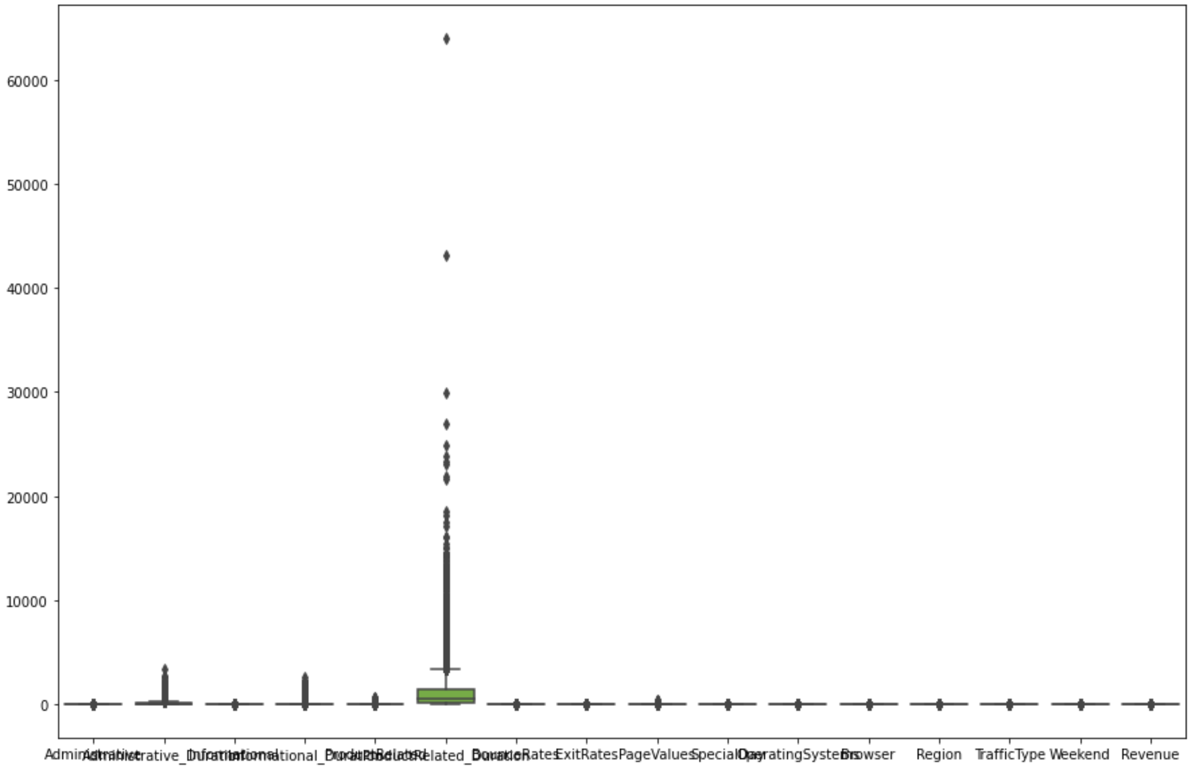
In [89]:

```
# Clean, wrangle, and handle missing data

print(df.describe())
```

```
       Administrative  Administrative_Duration  Informational  \
count    12330.000000             12330.000000   12202.000000   
mean         2.315166                80.818611       0.503770   
std          3.321784               176.779107       1.270882   
min          0.000000                 0.000000       0.000000   
25%          0.000000                 0.000000       0.000000   
50%          1.000000                 7.500000       0.000000   
75%          4.000000                93.256250       0.000000   
max         27.000000              3398.750000      24.000000   

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
count            12330.000000    12330.000000             12330.000000   
mean                34.472398       31.731468              1194.746220   
std                140.749294       44.475503              1913.669288   
min                  0.000000        0.000000                 0.000000   
25%                  0.000000        7.000000               184.137500   
50%                  0.000000       18.000000               598.936905   
75%                  0.000000       38.000000              1464.157214   
max               2549.375000      705.000000             63973.522230   

        BounceRates     ExitRates    PageValues    SpecialDay  \
count  12330.000000  12330.000000  12195.000000  12330.000000   
mean       0.022191      0.043073      5.911196      0.061427   
std        0.048488      0.048597     18.632116      0.198917   
min        0.000000      0.000000      0.000000      0.000000   
25%        0.000000      0.014286      0.000000      0.000000   
50%        0.003112      0.025156      0.000000      0.000000   
75%        0.016813      0.050000      0.000000      0.000000   
max        0.200000      0.200000    361.763742      1.000000   

       OperatingSystems       Browser        Region   TrafficType  
count      12207.000000  12330.000000  12330.000000  12330.000000  
mean           2.123618      2.357097      3.147364      4.069586  
std            0.911829      1.717277      2.401591      4.025169  
min            1.000000      1.000000      1.000000      1.000000  
25%            2.000000      2.000000      1.000000      2.000000  
50%            2.000000      2.000000      3.000000      2.000000  
75%            3.000000      2.000000      4.000000      4.000000  
max            8.000000     13.000000      9.000000     20.000000  
```

In [90]:

```
import matplotlib.pyplot as plt
import seaborn as sns

#visualizing outliers with boxplot
plt.figure(figsize = (15,10))
ax = sns.boxplot(data=df)
```



In [91]:

```
# replace outliers with mean Administrative
df['Administrative'].values[df['Administrative'] >= 10] = 2.315166
print(df['Administrative'].describe())
```

```
count    12330.000000
mean         1.837388
std          2.382258
min          0.000000
25%          0.000000
50%          1.000000
75%          3.000000
max          9.000000
Name: Administrative, dtype: float64
```

In [93]:

```
#replace outliers with mean Administrative_Duration
Q3 = 93.256250
UpperBoundary = Q3*1.5

df['Administrative_Duration'].values[df['Administrative_Duration'] >= UpperBoundary] = 80.
818611
print(df['Administrative_Duration'].describe())
```

```
count    12330.000000
mean        34.374434
std         40.305343
min          0.000000
25%          0.000000
50%          7.500000
75%         80.818611
max        139.681818
Name: Administrative_Duration, dtype: float64
```

In [94]:

```
# wrangle Informational
print(df['Informational'].describe())
```

```
count    12202.000000
mean         0.503770
std          1.270882
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max         24.000000
Name: Informational, dtype: float64
```

In [95]:

```
df['Informational'].values[df['Informational'] >= 0] = 0
print(df['Informational'].describe())
```

```
count    12202.0
mean         0.0
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: Informational, dtype: float64
```

In [96]:

```
#fill in missing values with 0
df['Informational'].fillna(0)
```

Out[96]:

```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
12325    0.0
12326    0.0
12327    0.0
12328    0.0
12329    0.0
Name: Informational, Length: 12330, dtype: float64
```

In [97]:

```
#wrangle Informational_Duration
df['Informational_Duration'].describe()
```

Out[97]:

```
count    12330.000000
mean        34.472398
std        140.749294
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max       2549.375000
Name: Informational_Duration, dtype: float64
```

In [98]:

```
df['Informational_Duration'].values[df['Informational_Duration'] >= 0] = 0
print(df['Informational_Duration'].describe())
```

```
count    12330.0
mean         0.0
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: Informational_Duration, dtype: float64
```

In [99]:

```
#wrangle Product_Related
Outlier = 38*1.5
df['ProductRelated'].values[df['ProductRelated'] >= Outlier] = 31.731468
print(df['ProductRelated'].describe())
```

```
count    12330.000000
mean        19.772587
std         13.871250
min          0.000000
25%          7.000000
50%         18.000000
75%         31.000000
max         56.000000
Name: ProductRelated, dtype: float64
```

In [100]:

```
#wrangle ProductRelated_Duration
Q3 = 1464.157214
Q1 = 184.1375
Outlier = (Q3-Q1)*1.5
df['ProductRelated_Duration'].values[df['ProductRelated_Duration'] >= Outlier] = 1194.7462
20
print(df['ProductRelated_Duration'])
```

```
0           0.000000
1          64.000000
2           0.000000
3           2.666667
4         627.500000
             ...    
12325    1783.791667
12326     465.750000
12327     184.250000
12328     346.000000
12329      21.250000
Name: ProductRelated_Duration, Length: 12330, dtype: float64
```

In [101]:

```
#describe wrangled data frame

df.describe()
```

Out[101]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRela
count	12330.000000	12330.000000	12202.0	12330.0	12330.000
mean	1.837388	34.374434	0.0	0.0	19.772
std	2.382258	40.305343	0.0	0.0	13.871
min	0.000000	0.000000	0.0	0.0	0.000
25%	0.000000	0.000000	0.0	0.0	7.000
50%	1.000000	7.500000	0.0	0.0	18.000
75%	3.000000	80.818611	0.0	0.0	31.000
max	9.000000	139.681818	0.0	0.0	56.000

In [102]:

```
df.to_csv('finalproject.csv')
```

In [103]:

```
import seaborn as sns
import statsmodels.api as sm
import pymysql.cursors
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import numpy as np
import os
```

In [104]:

```
histogram = list(df.describe())
```

In [105]:

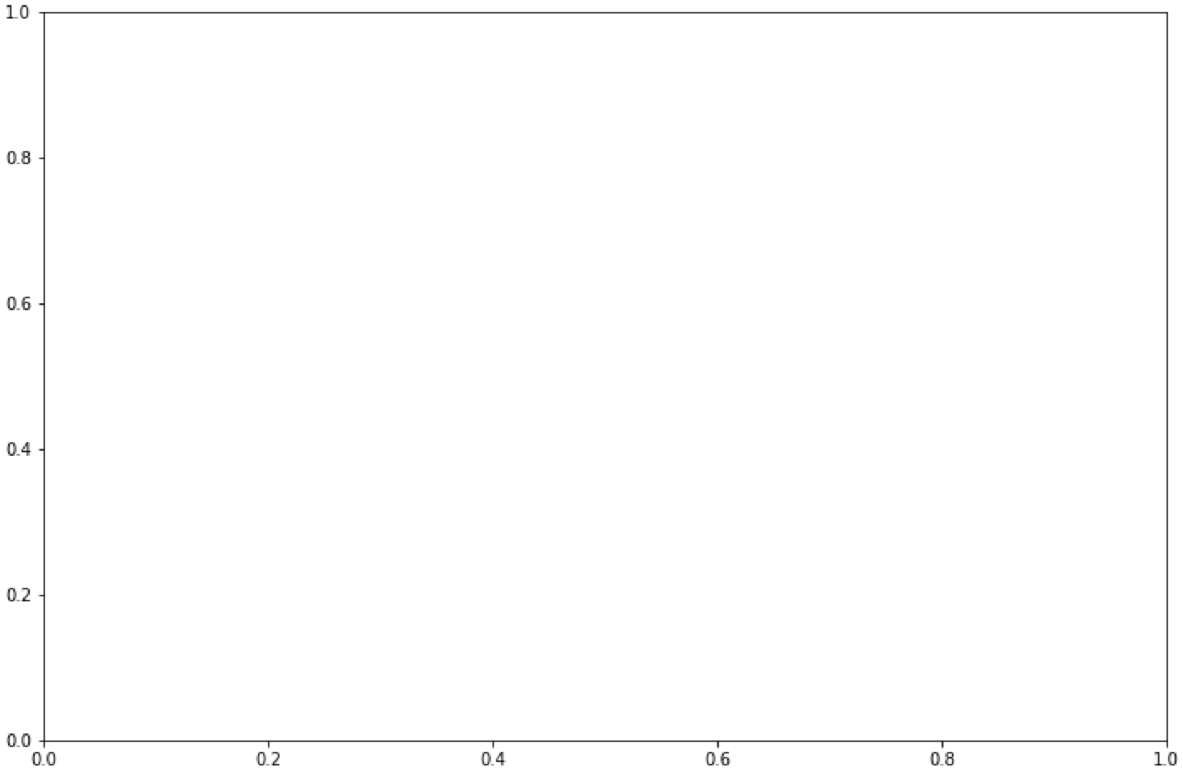
```
corr = df.corr().round(2)
```

In [106]:

```
mask = np.triu(np.ones_like(corr, dtype = bool))
```

In [107]:

```
f, ax = plt.subplots(figsize = (12,8))
```



In [108]:

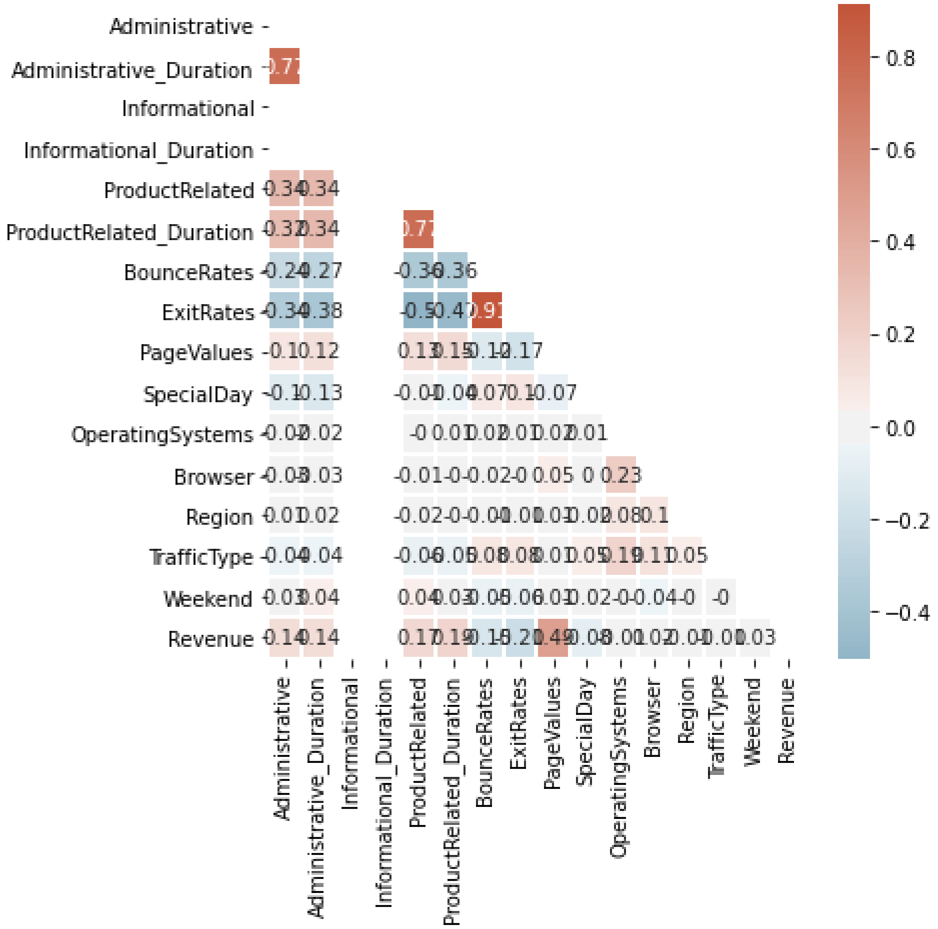
```
cmap = sns.diverging_palette(230,20,as_cmap = True)
```

In [109]:

```
sns.heatmap(corr, mask = mask, annot = True, center =0, linewidths = 1, cmap = cmap)
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2428e9c9bb0>
```



In [110]:

```
((df['Month'].groupby([df['VisitorType'], df['Revenue'],df['SpecialDay']]).count())/(df['V
isitorType'].count())).round(4)*100
```

Out[110]:

```
VisitorType        Revenue  SpecialDay
New_Visitor        False    0.0            9.98
                            0.2            0.03
                            0.4            0.05
                            0.6            0.15
                            0.8            0.06
                            1.0            0.05
                   True     0.0            3.29
                            0.2            0.04
                            0.4            0.03
                            0.6            0.03
                            0.8            0.01
                            1.0            0.02
Other              False    0.0            0.56
                   True     0.0            0.13
Returning_Visitor  False    0.0           64.47
                            0.2            1.30
                            0.4            1.82
                            0.6            2.47
                            0.8            2.48
                            1.0            1.12
                   True     0.0           11.43
                            0.2            0.07
                            0.4            0.07
                            0.6            0.20
                            0.8            0.08
                            1.0            0.06
Name: Month, dtype: float64
```

In [111]:

```
df.drop(df[df['ProductRelated_Duration'] > 10000].index, inplace = True)
```

In [112]:

```
    (df[['Administrative_Duration','Informational_Duration','ProductRelated_Duration']].groupby([df['Weekend'], df['Revenue']]).mean()/60).round(2)
```

Out[112]:

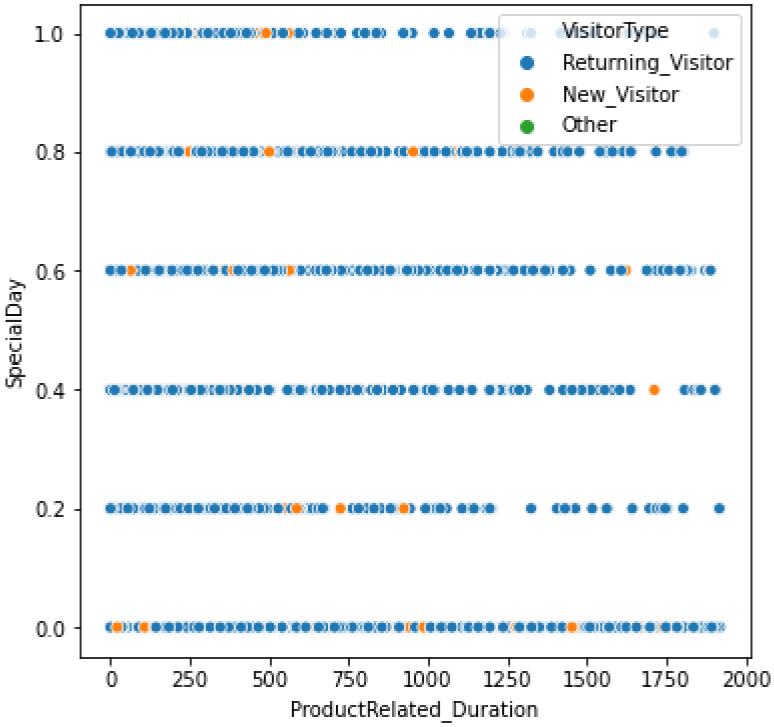
		Administrative_Duration	Informational_Duration	ProductRelated_Duration
Weekend	**Revenue**			
False	**False**	0.51	0.0	10.52
	True	0.80	0.0	15.29
True	**False**	0.59	0.0	11.11
	True	0.80	0.0	15.36

In [113]:

```
sns.scatterplot(data = df, x ="ProductRelated_Duration", y = "SpecialDay", hue = "VisitorType")
```

Out[113]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2428eb41340>
```



In [114]:

```
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

In [115]:

```
le = LabelEncoder()
df['Revenue'] = le.fit_transform(df['Revenue'])
df['Month'] = le.fit_transform(df['Month'])
df['Weekend'] = le.fit_transform(df['Weekend'])
df['VisitorType'] = le.fit_transform(df['VisitorType'])
#f['Weekend'].value_counts()
```

In [60]:

```
y=df.Revenue
x=df.drop('Revenue',axis=1)
```

In [61]:

```
### 70% training data set
### 30% Test data set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3, random_state = 0)
x_train.head()
```

Out[61]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRela
11332	1	7.125000	0.0	0.0	
12071	0	0.000000	0.0	0.0	
10023	0	0.000000	0.0	0.0	
6771	9	80.818611	0.0	0.0	
4283	0	0.000000	0.0	0.0	

In [62]:

```
print('Training Features Shape:', x_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', x_test.shape)
print('Testing Labels Shape:', y_test.shape)
```

```
Training Features Shape: (8631, 17)
Training Labels Shape: (8631,)
Testing Features Shape: (3699, 17)
Testing Labels Shape: (3699,)
```

In [74]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
model = RandomForestClassifier()
```

In [75]:

```
#df1 = x_train[['Informational','ExitRates', 'PageValues']]
x_train = x_train[['Administrative','Administrative_Duration','Informational_Duration','ProductRelated','ProductRelated_Duration','BounceRates','SpecialDay','Month','Weekend',
                   'Browser','Region','TrafficType','VisitorType','Weekend']]
x_test = x_test[['Administrative','Administrative_Duration','Informational_Duration','ProductRelated','ProductRelated_Duration',
                 'BounceRates','SpecialDay','Month','Weekend','Browser','Region','TrafficType','VisitorType','Weekend']]
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

In [76]:

```
# evaluating the model
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
```

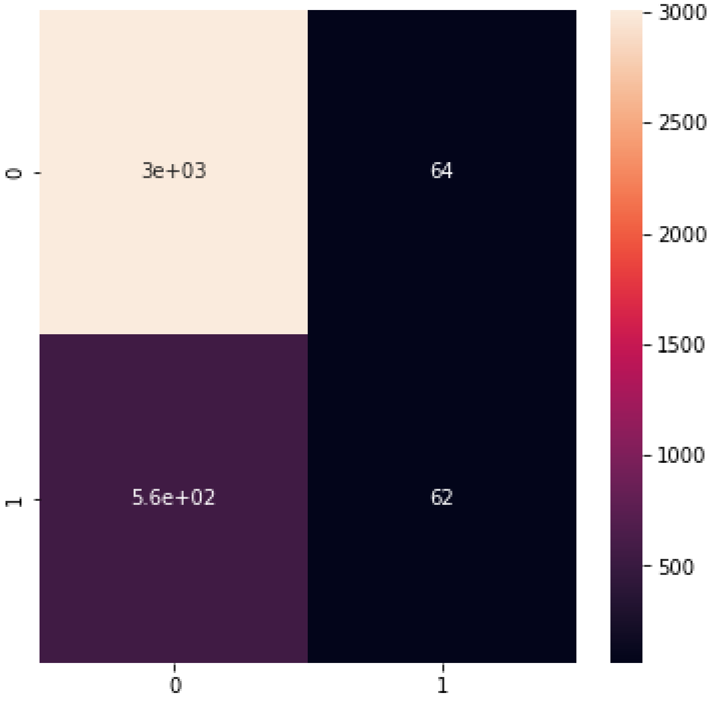
```
Training Accuracy : 0.9996524157108099
Testing Accuracy : 0.8313057583130575
```

In [80]:

```
# confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.rcParams['figure.figsize'] = (6, 6)
sns.heatmap(cm ,annot = True)
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2428d9194c0>
```



In [69]:

```
# classification report
cr = classification_report(y_test, y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.84      0.98      0.91      3077
           1       0.49      0.10      0.17       622

    accuracy                           0.83      3699
   macro avg       0.67      0.54      0.54      3699
weighted avg       0.78      0.83      0.78      3699
```