EDA- Data preparation Online shoppers

November 7, 2021

1 EDA- Data preparation

[1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
#!pip install featurewiz
from featurewiz import featurewiz
```

```
Imported featurewiz: advanced feature engg and selection library. Version=0.0.42
output = featurewiz(dataname, target, corr_limit=0.70,
                    verbose=2, sep=',', header=0, test_data='',
                    feature_engg='', category_encoders='')
Create new features via 'feature_engg' flag :
['interactions','groupby','target']
```

1.1 Import Dataset

[2]:
```
df =pd.read_csv ('/home/jovyan/online_shoppers_intention.csv')
```

1.2 Dataset Description

[3]:
```
print (df)
```

```
       Administrative  Administrative_Duration  Informational  \
0                   0                      0.0              0
1                   0                      0.0              0
2                   0                      0.0              0
3                   0                      0.0              0
4                   0                      0.0              0
…                 …                      …            …
12325               3                    145.0              0
12326               0                      0.0              0
12327               0                      0.0              0
```

```
12328              4                    75.0            0
12329              0                     0.0            0

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
0                         0.0               1                 0.000000
1                         0.0               2                64.000000
2                         0.0               1                 0.000000
3                         0.0               2                 2.666667
4                         0.0              10               627.500000
…                        …             …                      …
12325                     0.0              53              1783.791667
12326                     0.0               5               465.750000
12327                     0.0               6               184.250000
12328                     0.0              15               346.000000
12329                     0.0               3                21.250000

       BounceRates  ExitRates  PageValues  SpecialDay Month  OperatingSystems  \
0         0.200000   0.200000    0.000000         0.0   Feb                 1
1         0.000000   0.100000    0.000000         0.0   Feb                 2
2         0.200000   0.200000    0.000000         0.0   Feb                 4
3         0.050000   0.140000    0.000000         0.0   Feb                 3
4         0.020000   0.050000    0.000000         0.0   Feb                 3
…              …          …          …         …   …                …
12325     0.007143   0.029031   12.241717         0.0   Dec                 4
12326     0.000000   0.021333    0.000000         0.0   Nov                 3
12327     0.083333   0.086667    0.000000         0.0   Nov                 3
12328     0.000000   0.021053    0.000000         0.0   Nov                 2
12329     0.000000   0.066667    0.000000         0.0   Nov                 3

       Browser  Region  TrafficType        VisitorType  Weekend  Revenue
0            1       1            1  Returning_Visitor    False    False
1            2       1            2  Returning_Visitor    False    False
2            1       9            3  Returning_Visitor    False    False
3            2       2            4  Returning_Visitor    False    False
4            3       1            4  Returning_Visitor     True    False
…          …     …          …                …      …      …
12325        6       1            1  Returning_Visitor     True    False
12326        2       1            8  Returning_Visitor     True    False
12327        2       1           13  Returning_Visitor     True    False
12328        2       3           11  Returning_Visitor    False    False
12329        2       1            2        New_Visitor     True    False

[12330 rows x 18 columns]
```

[4]:
```
df.head(5)
```

[4]:

```
   Administrative  Administrative_Duration  Informational  \
0               0                      0.0              0
1               0                      0.0              0
2               0                      0.0              0
3               0                      0.0              0
4               0                      0.0              0

   Informational_Duration  ProductRelated  ProductRelated_Duration  \
0                     0.0               1                 0.000000
1                     0.0               2                64.000000
2                     0.0               1                 0.000000
3                     0.0               2                 2.666667
4                     0.0              10               627.500000

   BounceRates  ExitRates  PageValues  SpecialDay Month  OperatingSystems  \
0         0.20       0.20         0.0         0.0   Feb                 1
1         0.00       0.10         0.0         0.0   Feb                 2
2         0.20       0.20         0.0         0.0   Feb                 4
3         0.05       0.14         0.0         0.0   Feb                 3
4         0.02       0.05         0.0         0.0   Feb                 3

   Browser  Region  TrafficType        VisitorType  Weekend  Revenue
0        1       1            1  Returning_Visitor    False    False
1        2       1            2  Returning_Visitor    False    False
2        1       9            3  Returning_Visitor    False    False
3        2       2            4  Returning_Visitor    False    False
4        3       1            4  Returning_Visitor     True    False
```

1.2.1 Data Types

[5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
```

```
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

Most of the dataset attributes are numerical, either integers or floats; Revenue(the class label) and Weekend are boolean type*

1.2.2 Statistical Analysis of the Dataset

[6]:
```
df.describe()
```

[6]:
```
       Administrative  Administrative_Duration  Informational  \
count    12330.000000             12330.000000   12330.000000
mean         2.315166                80.818611       0.503569
std          3.321784               176.779107       1.270156
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              1464.157213
max               2549.375000      705.000000             63973.522230

        BounceRates     ExitRates    PageValues    SpecialDay  \
count  12330.000000  12330.000000  12330.000000  12330.000000
mean       0.022191      0.043073      5.889258      0.061427
std        0.048488      0.048597     18.568437      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
```

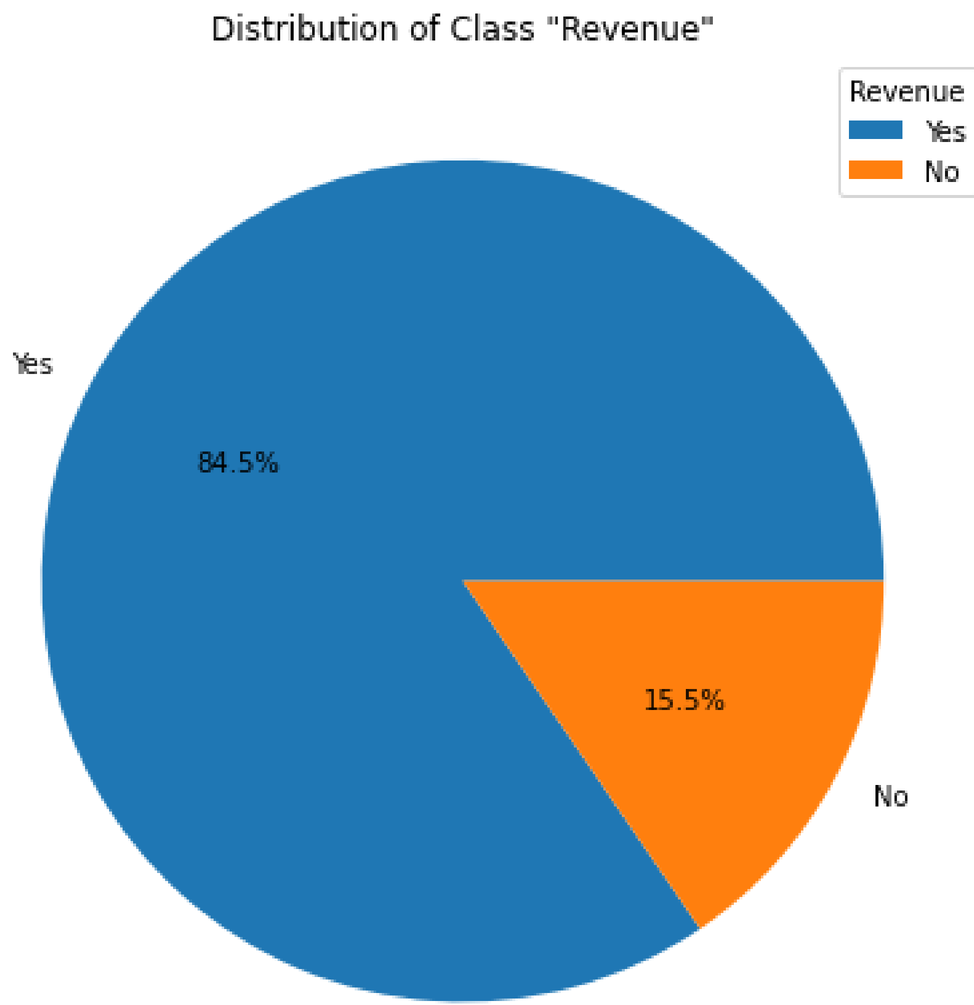
```
max        0.200000      0.200000    361.763742      1.000000

       OperatingSystems       Browser        Region   TrafficType
count      12330.000000  12330.000000  12330.000000  12330.000000
mean           2.124006      2.357097      3.147364      4.069586
std            0.911325      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
```

1.2.3 Count value of Revenue attribute to verify imbalance

[7]:
```python
df['Revenue'].value_counts()
```

[7]:
```
False    10422
True      1908
Name: Revenue, dtype: int64
```

[8]:
```python
x = df['Revenue'].value_counts()
labels =  ["Yes", "No"]

fig, ax = plt.subplots(figsize=(6, 6))
ax.pie(x, labels=labels, autopct='%.1f%%')
ax.set_title('Distribution of Class "Revenue"')
plt.tight_layout()
plt.legend(title = "Revenue")
plt.savefig('output.png', dpi=300, bbox_inches='tight')
```



1.2.4 Missing values in Dataset

[9]:
```
df.isna().sum()
```

[9]:
```
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
```

```
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

This dataset doesn't have missing values, no actions are needed

1.2.5 Data type transformation

[10]:
```python
df.Revenue = df.Revenue.astype('int')
df.Weekend = df.Weekend.astype('int')
```

The attributes Revenue and Weekend were changed from boolean into binary, use them more easily for calculations

[11]:
```python
d_month = {'Jan':1,'Feb':2, 'Mar':3,'Apr':4, 'May':5, 'June':6, 'Jul':7, 'Aug':
 ↪8, 'Sep':9, 'Oct':10,'Nov':11, 'Dec':12}
d_visitor = {'Returning_Visitor': 1,'New_Visitor':2, 'Other':3}
df.Month = df.Month.map(d_month)
df.VisitorType = df.VisitorType.map(d_visitor)
```

The attributes Month and Visitor were changed from object into int

1.2.6 Dataset with modified values

[12]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
```

```
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  int64
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  int64
 16  Weekend                  12330 non-null  int64
 17  Revenue                  12330 non-null  int64
dtypes: float64(7), int64(11)
memory usage: 1.7 MB
```

1.2.7 Correlation Analysis

[13]:
```python
data_correlation = df.corr()
print(data_correlation)
```

```
                         Administrative  Administrative_Duration  \
Administrative                 1.000000                 0.601583
Administrative_Duration        0.601583                 1.000000
Informational                  0.376850                 0.302710
Informational_Duration         0.255848                 0.238031
ProductRelated                 0.431119                 0.289087
ProductRelated_Duration        0.373939                 0.355422
BounceRates                   -0.223563                -0.144170
ExitRates                     -0.316483                -0.205798
PageValues                     0.098990                 0.067608
SpecialDay                    -0.094778                -0.073304
Month                          0.096713                 0.057885
OperatingSystems              -0.006347                -0.007343
Browser                       -0.025035                -0.015392
Region                        -0.005487                -0.005561
TrafficType                   -0.033561                -0.014376
VisitorType                    0.016680                 0.019120
Weekend                        0.026417                 0.014990
Revenue                        0.138917                 0.093587

                         Informational  Informational_Duration  \
Administrative                0.376850                0.255848
Administrative_Duration       0.302710                0.238031
Informational                 1.000000                0.618955
Informational_Duration        0.618955                1.000000
ProductRelated                0.374164                0.280046
ProductRelated_Duration       0.387505                0.347364
```

```
BounceRates                  -0.116114                -0.074067
ExitRates                    -0.163666                -0.105276
PageValues                    0.048632                 0.030861
SpecialDay                   -0.048219                -0.030577
Month                         0.063500                 0.044354
OperatingSystems             -0.009527                -0.009579
Browser                      -0.038235                -0.019285
Region                       -0.029169                -0.027144
TrafficType                  -0.034491                -0.024675
VisitorType                  -0.058211                -0.045372
Weekend                       0.035785                 0.024078
Revenue                       0.095200                 0.070345

                         ProductRelated  ProductRelated_Duration  BounceRates  \
Administrative                 0.431119                 0.373939    -0.223563
Administrative_Duration        0.289087                 0.355422    -0.144170
Informational                  0.374164                 0.387505    -0.116114
Informational_Duration         0.280046                 0.347364    -0.074067
ProductRelated                 1.000000                 0.860927    -0.204578
ProductRelated_Duration        0.860927                 1.000000    -0.184541
BounceRates                   -0.204578                -0.184541     1.000000
ExitRates                     -0.292526                -0.251984     0.913004
PageValues                     0.056282                 0.052823    -0.119386
SpecialDay                    -0.023958                -0.036380     0.072702
Month                          0.156022                 0.137520    -0.066562
OperatingSystems               0.004290                 0.002976     0.023823
Browser                       -0.013146                -0.007380    -0.015772
Region                        -0.038122                -0.033091    -0.006485
TrafficType                   -0.043064                -0.036377     0.078286
VisitorType                   -0.127916                -0.118273    -0.114916
Weekend                        0.016092                 0.007311    -0.046514
Revenue                        0.158538                 0.152373    -0.150673

                         ExitRates  PageValues  SpecialDay     Month  \
Administrative           -0.316483    0.098990   -0.094778  0.096713
Administrative_Duration  -0.205798    0.067608   -0.073304  0.057885
Informational            -0.163666    0.048632   -0.048219  0.063500
Informational_Duration   -0.105276    0.030861   -0.030577  0.044354
ProductRelated           -0.292526    0.056282   -0.023958  0.156022
ProductRelated_Duration  -0.251984    0.052823   -0.036380  0.137520
BounceRates               0.913004   -0.119386    0.072702 -0.066562
ExitRates                 1.000000   -0.174498    0.102242 -0.095465
PageValues               -0.174498    1.000000   -0.063541  0.067198
SpecialDay                0.102242   -0.063541    1.000000 -0.256901
Month                    -0.095465    0.067198   -0.256901  1.000000
OperatingSystems          0.014567    0.018508    0.012652  0.038407
Browser                  -0.004442    0.045592    0.003499  0.020120
Region                   -0.008907    0.011315   -0.016098  0.023894
```

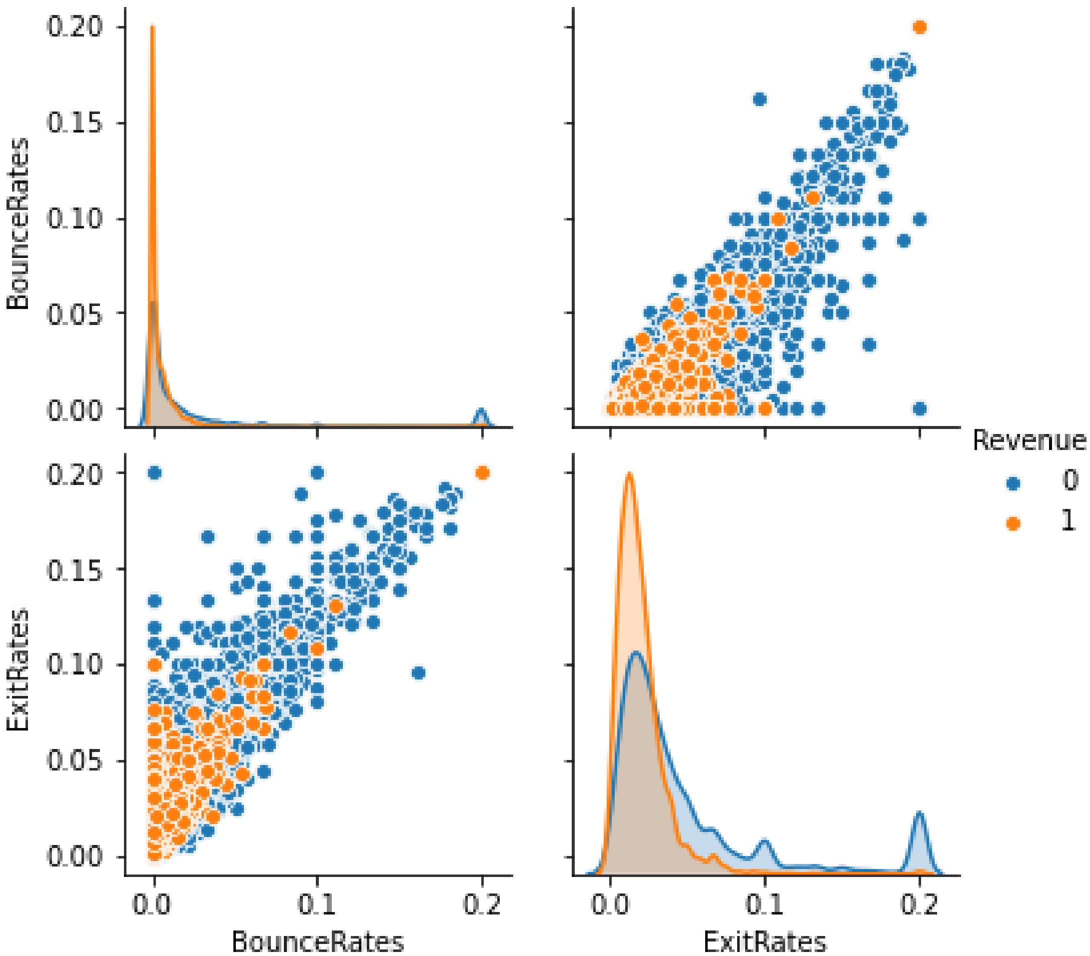
```
TrafficType               0.078616   0.012532   0.052301  0.054941
VisitorType              -0.152678   0.120077  -0.086854  0.126110
Weekend                  -0.062587   0.012002  -0.016767  0.017150
Revenue                  -0.207071   0.492569  -0.082305  0.127372

                         OperatingSystems   Browser    Region  TrafficType  \
Administrative                  -0.006347 -0.025035 -0.005487    -0.033561
Administrative_Duration         -0.007343 -0.015392 -0.005561    -0.014376
Informational                   -0.009527 -0.038235 -0.029169    -0.034491
Informational_Duration          -0.009579 -0.019285 -0.027144    -0.024675
ProductRelated                   0.004290 -0.013146 -0.038122    -0.043064
ProductRelated_Duration          0.002976 -0.007380 -0.033091    -0.036377
BounceRates                      0.023823 -0.015772 -0.006485     0.078286
ExitRates                        0.014567 -0.004442 -0.008907     0.078616
PageValues                       0.018508  0.045592  0.011315     0.012532
SpecialDay                       0.012652  0.003499 -0.016098     0.052301
Month                            0.038407  0.020120  0.023894     0.054941
OperatingSystems                 1.000000  0.223013  0.076775     0.189154
Browser                          0.223013  1.000000  0.097393     0.111938
Region                           0.076775  0.097393  1.000000     0.047520
TrafficType                      0.189154  0.111938  0.047520     1.000000
VisitorType                      0.109981  0.124456  0.075819     0.068113
Weekend                          0.000284 -0.040261 -0.000691    -0.002221
Revenue                         -0.014668  0.023984 -0.011595    -0.005113

                         VisitorType   Weekend   Revenue
Administrative              0.016680  0.026417  0.138917
Administrative_Duration     0.019120  0.014990  0.093587
Informational              -0.058211  0.035785  0.095200
Informational_Duration     -0.045372  0.024078  0.070345
ProductRelated             -0.127916  0.016092  0.158538
ProductRelated_Duration    -0.118273  0.007311  0.152373
BounceRates                -0.114916 -0.046514 -0.150673
ExitRates                  -0.152678 -0.062587 -0.207071
PageValues                  0.120077  0.012002  0.492569
SpecialDay                 -0.086854 -0.016767 -0.082305
Month                       0.126110  0.017150  0.127372
OperatingSystems            0.109981  0.000284 -0.014668
Browser                     0.124456 -0.040261  0.023984
Region                      0.075819 -0.000691 -0.011595
TrafficType                 0.068113 -0.002221 -0.005113
VisitorType                 1.000000  0.030262  0.098485
Weekend                     0.030262  1.000000  0.029295
Revenue                     0.098485  0.029295  1.000000
```

[14]:
```
g1 = sns.pairplot(df[['Administrative', 'Informational', 'ProductRelated',␣
 ↪'PageValues', 'Region', 'SpecialDay','Revenue']], hue='Revenue')
```

```
g1.fig.suptitle('Attribues Correlations')
plt.savefig('output.png', dpi=300, bbox_inches='tight')
plt.show()
```

Correlation of Numerical attributes with Revenue

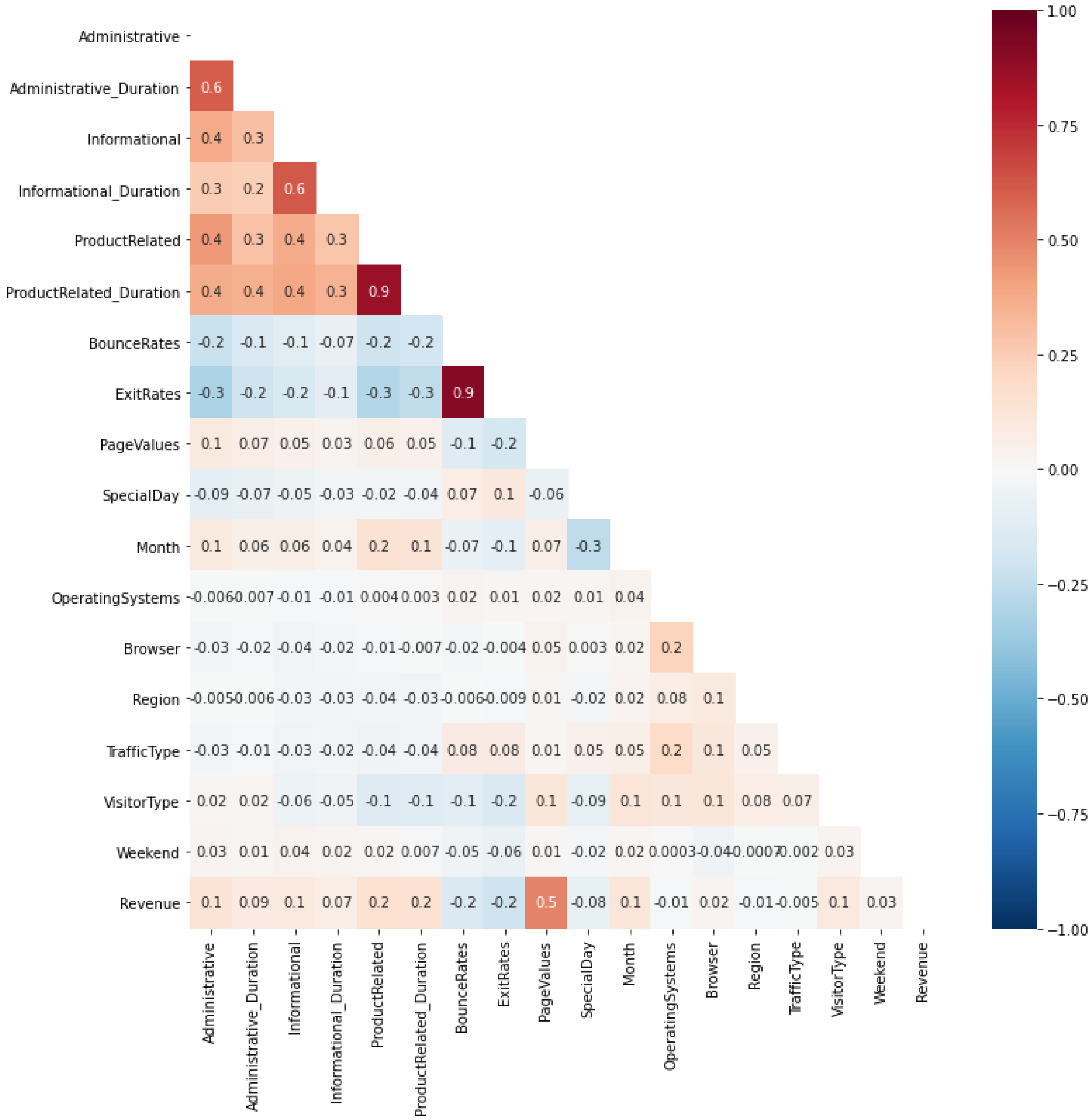
[15]:
```
sns.
↪pairplot(df,x_vars=['BounceRates','ExitRates'],y_vars=['BounceRates','ExitRates'],hue='Reve
plt.show()
```



Correlation of attributes BounceRates and Exit Rates with Revenue

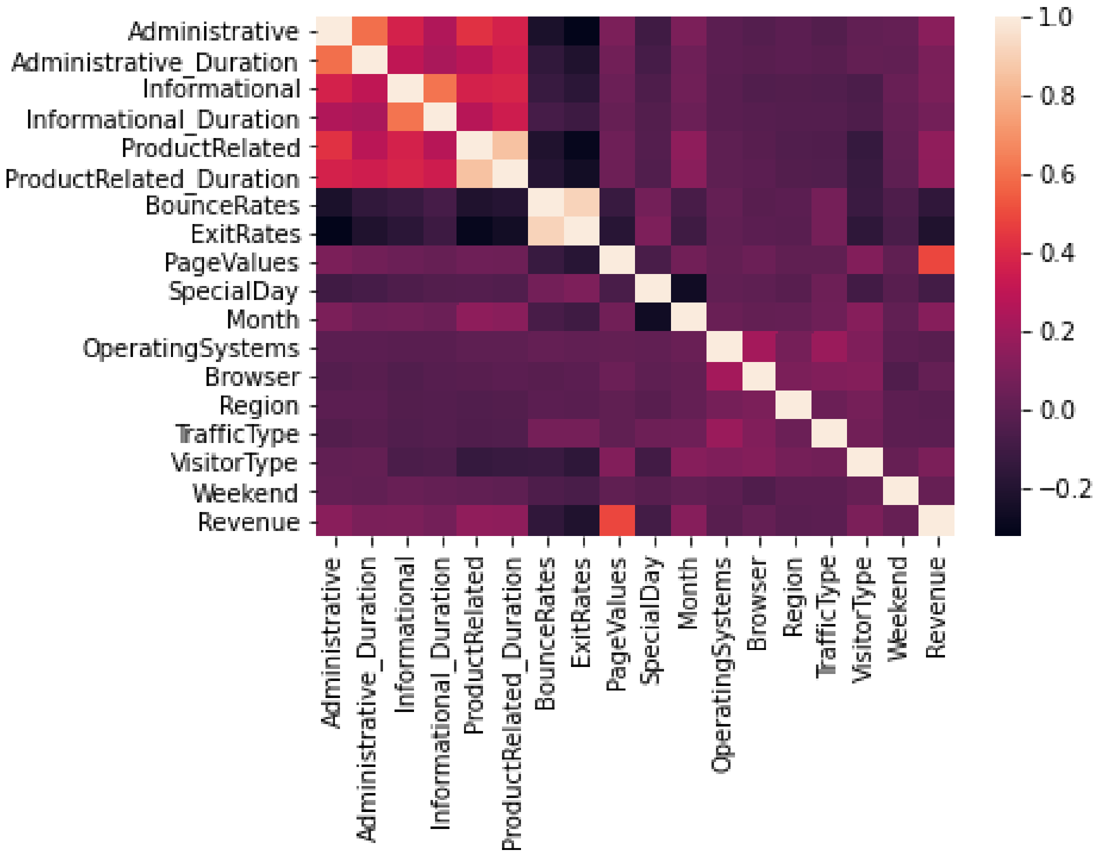
1.2.8 Correlation Matrix

[16]:
```
matrix = np.triu(data_correlation)
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(df.corr(), annot=True, ax=ax, fmt='.1g', vmin=-1, vmax=1, center=␣
 ↪0, mask=matrix, cmap='RdBu_r')
plt.savefig('output2.png', dpi=300, bbox_inches='tight')
plt.show()
```



[17]:
```
corr =data_correlation
sns.heatmap(corr,
xticklabels=corr.columns.values,
yticklabels=corr.columns.values)
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9c49fb7890>



From the above correlation analysis we can infer the following:

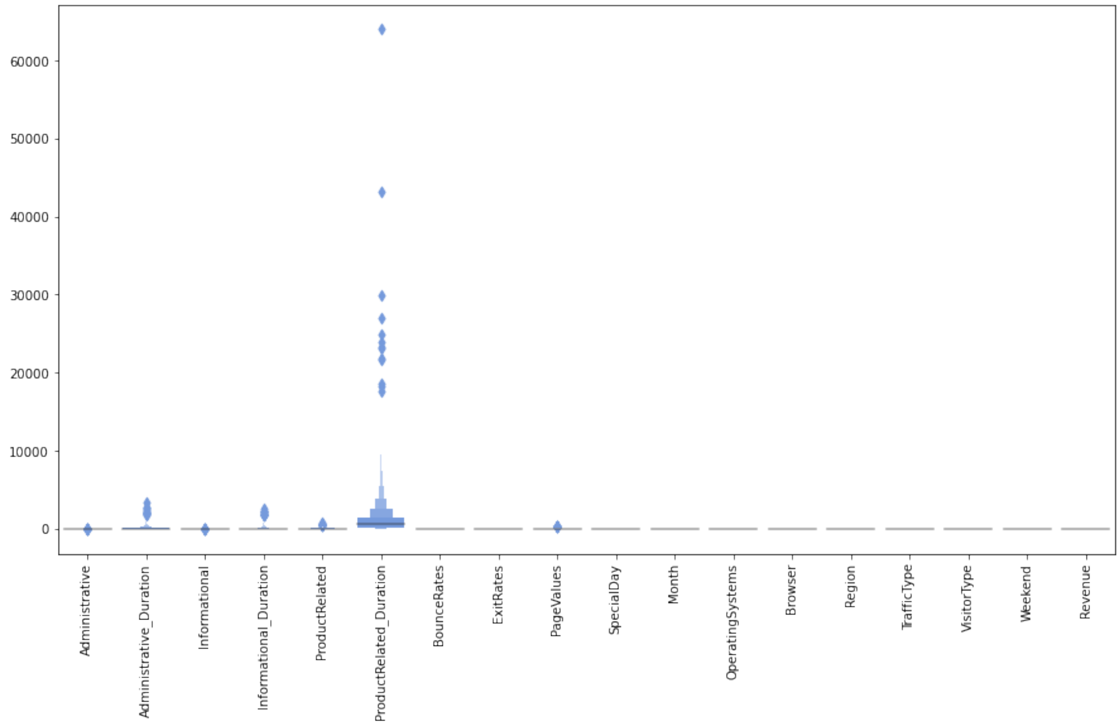
- There are only a couple of attributes that have a high correlation of 0.9 with eachother:
  - ProductRelated and ProductRelated_Duration
  - ExitRates and BounceRates
- In addition, the following attributes have a moderate correlation:
  - PageValues and Revenue have a correlation of 0.5
  - Administrative and Administrative_Duration & Informational and Informational_Duration have a correlation of 0.6

1.3 Boxplot of attributes

[18]:
```
#plt.boxplot(df)

# show plot
#plt.show()
```

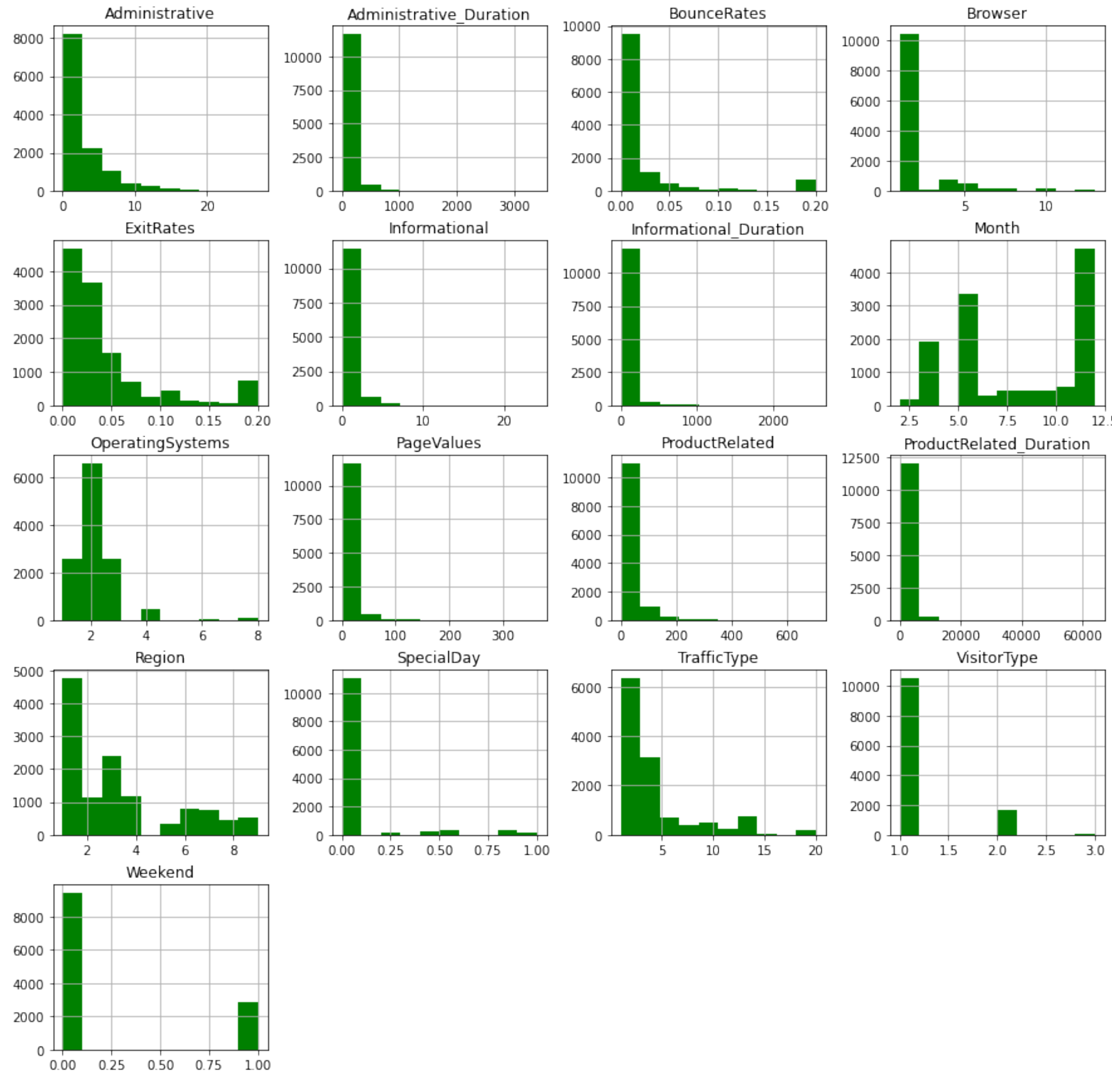
[19]:
```
plt.figure(figsize=(15,8))
sns.boxenplot(data = df,color = "cornflowerblue")
```

```
plt.xticks(rotation=90)
plt.savefig('output3.png', dpi=300, bbox_inches='tight')
plt.show()
```



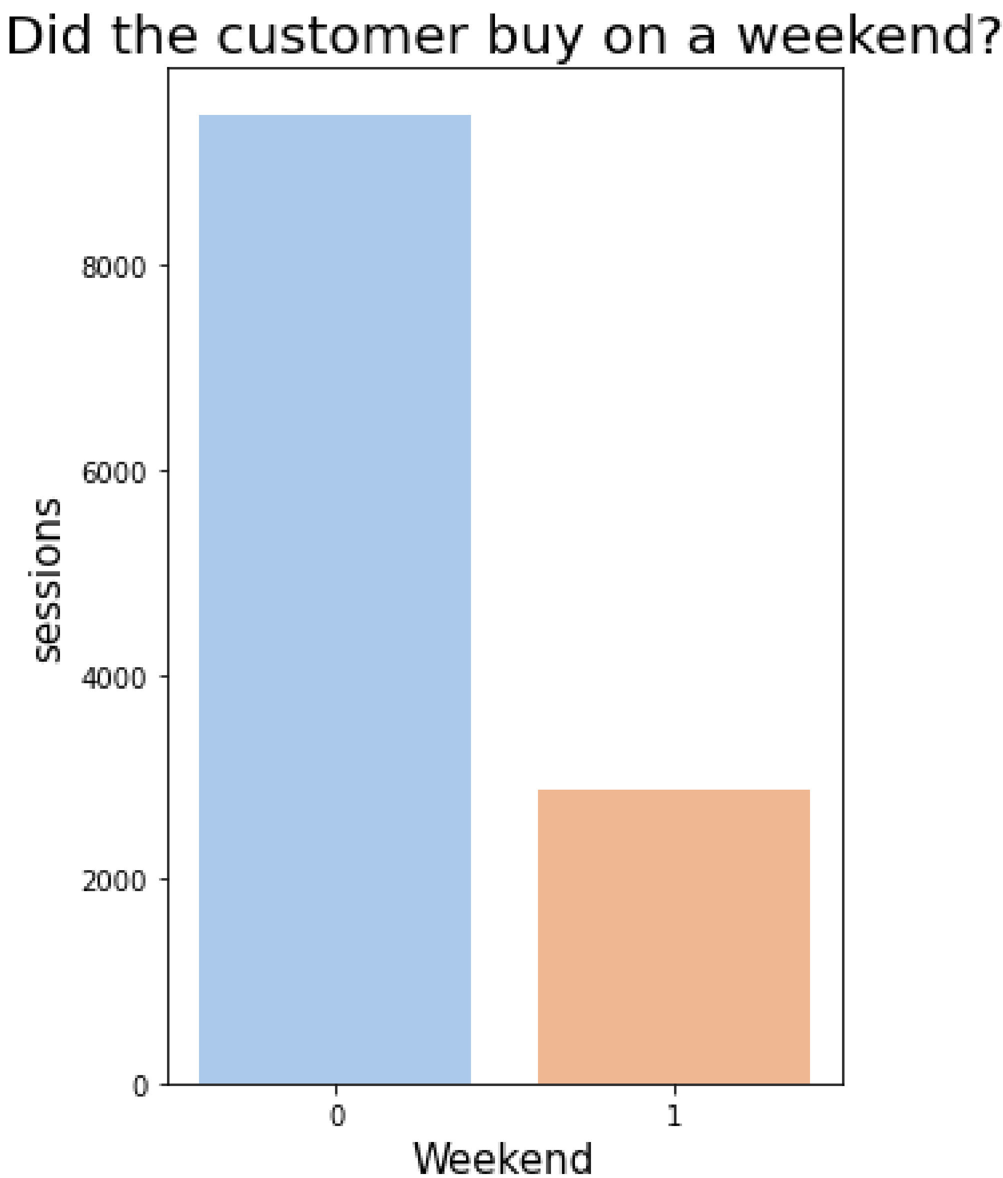
1.4 Histogram of attributes

[20]:
```
df_X = df.drop(columns=['Revenue'])
fig, ax = plt.subplots(1, 1, figsize=(15, 15))
df_X.hist(ax=ax, color = 'green')
plt.savefig('output4.png', dpi=300, bbox_inches='tight')
plt.show()
```

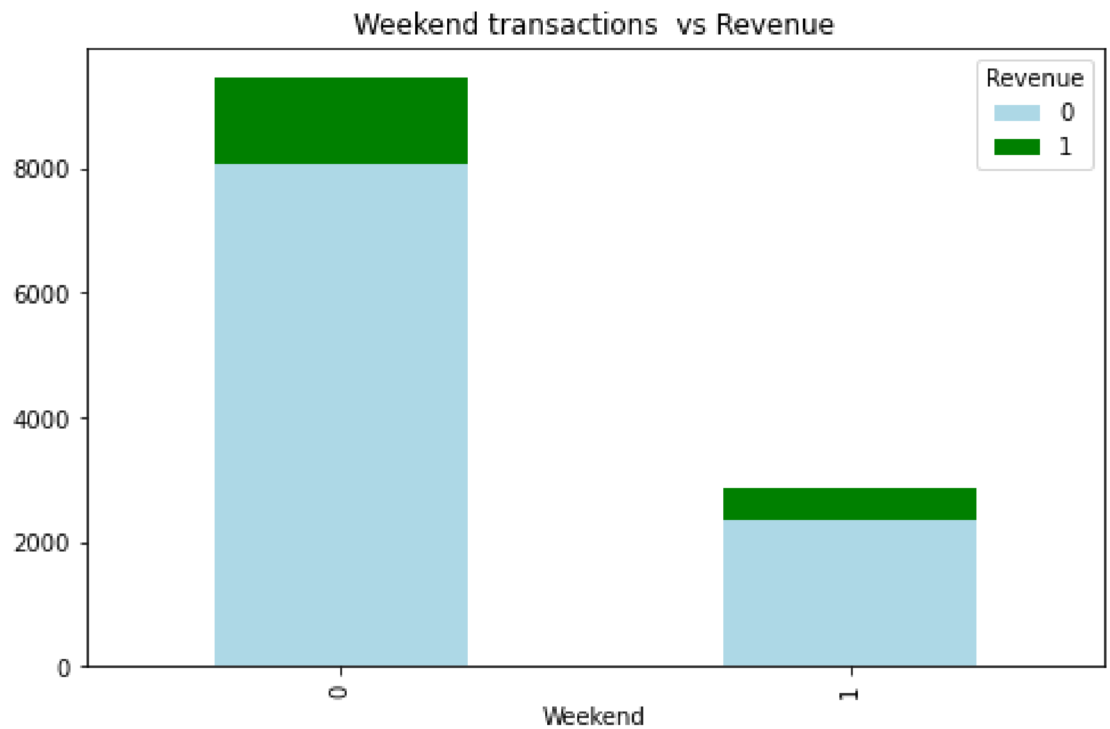


1.5 Distribution of customers activity on a Weekend

[21]:
```
plt.figure(figsize = (10,7))

plt.subplot(1, 2, 1)
sns.countplot(df['Weekend'], palette = 'pastel')
plt.title('Did the customer buy on a weekend?', fontsize = 20)
plt.xlabel('Weekend', fontsize = 15)
plt.ylabel('sessions', fontsize = 15)
plt.savefig('output5.png', dpi=300, bbox_inches='tight')
plt.show()
```



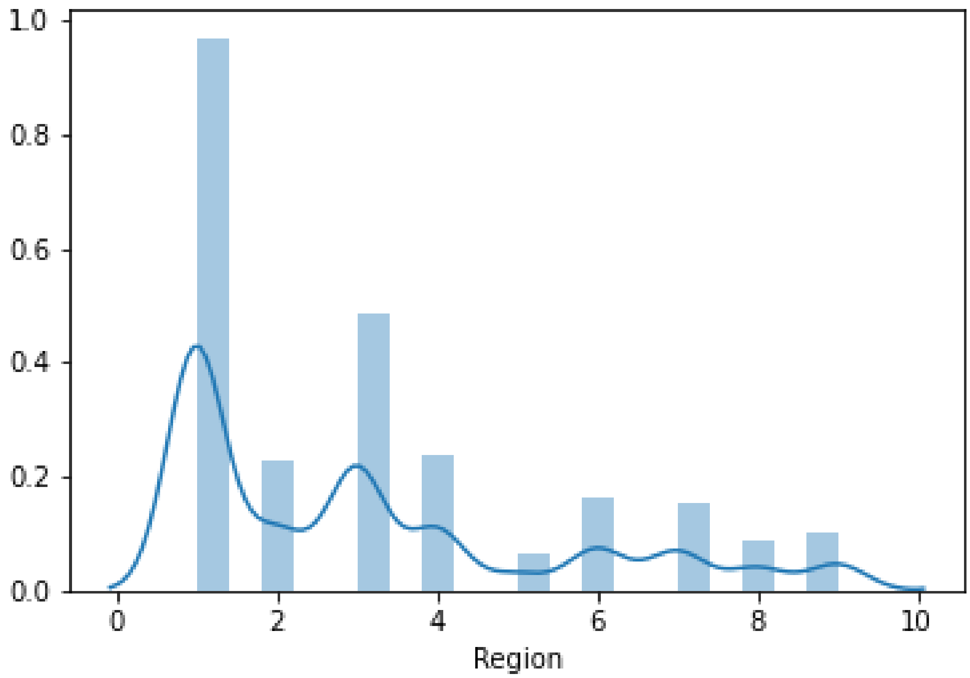
[22]:
```
plot1 = pd.crosstab(df['Weekend'], df['Revenue'])
plot1.plot(kind = 'bar', stacked = True, figsize = (8, 5), color =␣
↪['lightblue', 'green'])
plt.title('Weekend transactions  vs Revenue')
plt.savefig('output6.png', dpi=300, bbox_inches='tight')
plt.show()
```



From the above graphs we can observe that the majority of sessions didn't happen on a weekend and how the revenue per weekend is reflected

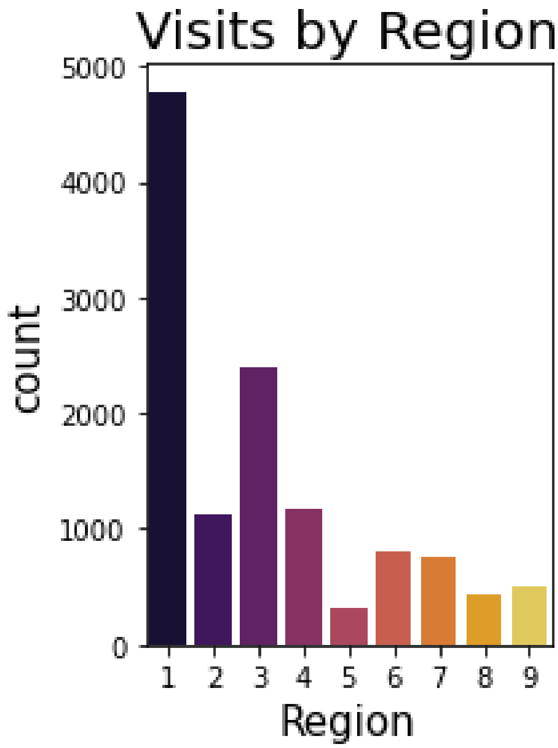
1.6 Distribution for Region attribute

[23]:
```
sns.distplot(df['Region'], bins=20)
plt.show()
```



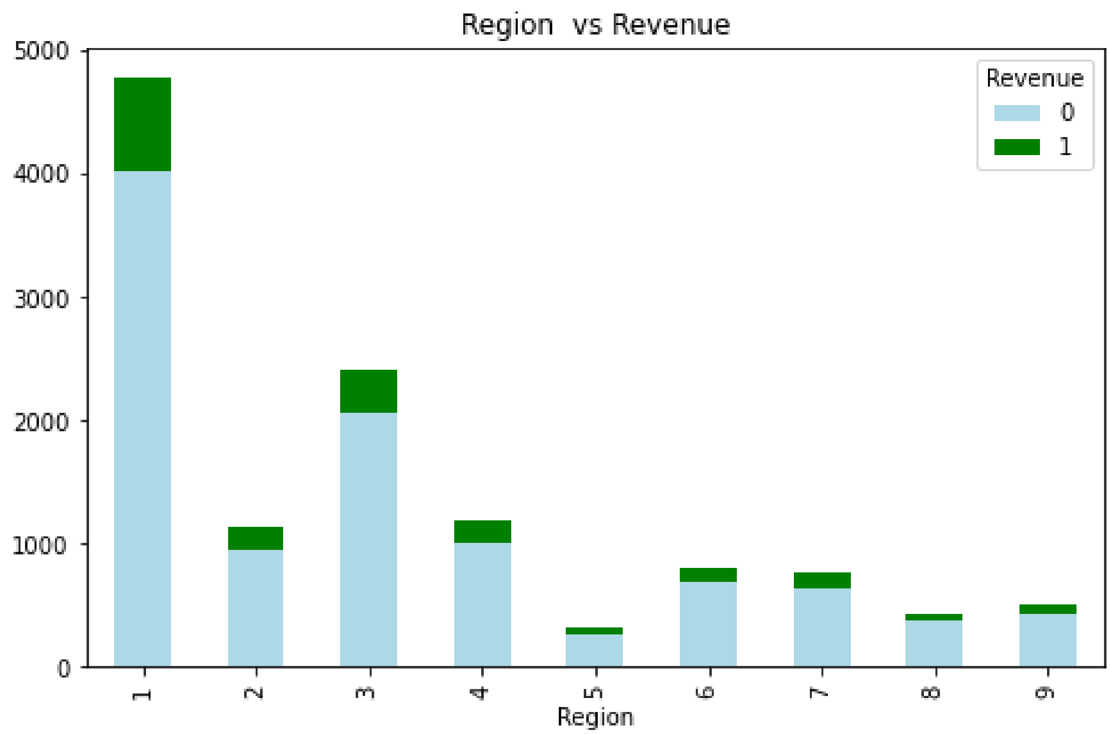
1.7 Checking the Distribution of Sales by Region

[24]:
```
plt.subplot(1, 2, 2)
sns.countplot(df['Region'], palette = 'inferno')
plt.title('Visits by Region', fontsize = 20)
plt.xlabel('Region', fontsize = 15)
plt.ylabel('count', fontsize = 15)
plt.savefig('output7.png', dpi=300, bbox_inches='tight')
```



1.8 Region vs revenue

[25]:
```
pps = pd.crosstab(df['Region'], df['Revenue'])
pps.plot(kind = 'bar', stacked = True, figsize = (8, 5), color = ['lightblue',␣
↪'green'])
plt.title('Region  vs Revenue')
plt.savefig('output8.png', dpi=300, bbox_inches='tight')
plt.show()
```

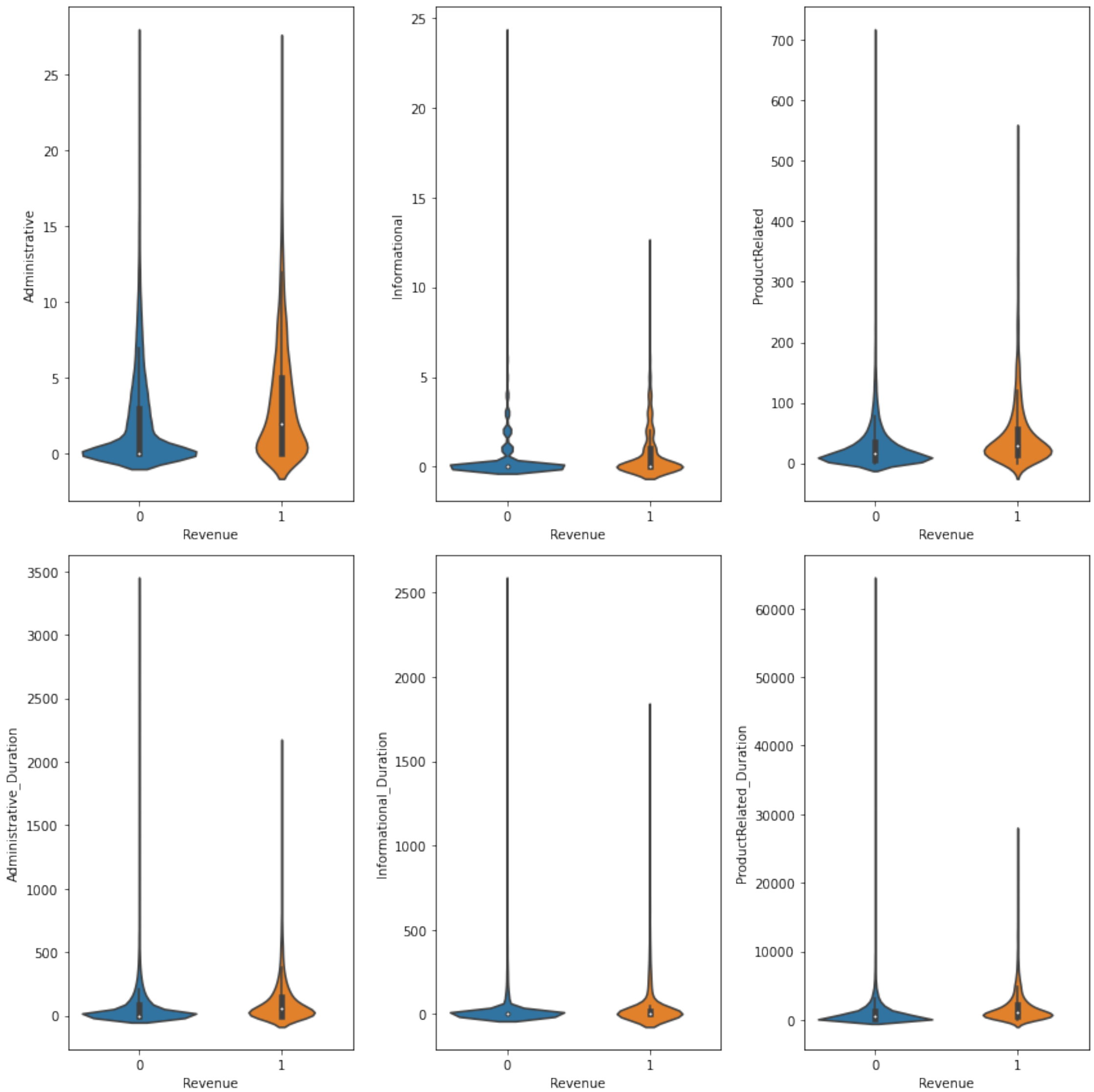


From the above graphs we can observe that the behavior of the sessions by Geographical Region and how it behaves vs Revenue

1.9 Web Pages Analysis

[26]:
```
fig = plt.figure(figsize=(12, 12))

ax1 = fig.add_subplot(2, 3, 1)
ax2 = fig.add_subplot(2, 3, 2)
ax3 = fig.add_subplot(2, 3, 3)
ax4 = fig.add_subplot(2, 3, 4)
ax5 = fig.add_subplot(2, 3, 5)
ax6 = fig.add_subplot(2, 3, 6)

sns.violinplot(data=df, x = 'Revenue', y = 'Administrative', ax=ax1)
sns.violinplot(data=df, x = 'Revenue', y = 'Informational', ax=ax2)
sns.violinplot(data=df, x = 'Revenue', y = 'ProductRelated', ax=ax3)
sns.violinplot(data=df, x = 'Revenue', y = 'Administrative_Duration', ax=ax4)
sns.violinplot(data=df, x = 'Revenue', y = 'Informational_Duration', ax=ax5)
sns.violinplot(data=df, x = 'Revenue', y = 'ProductRelated_Duration', ax=ax6)

plt.tight_layout()
```

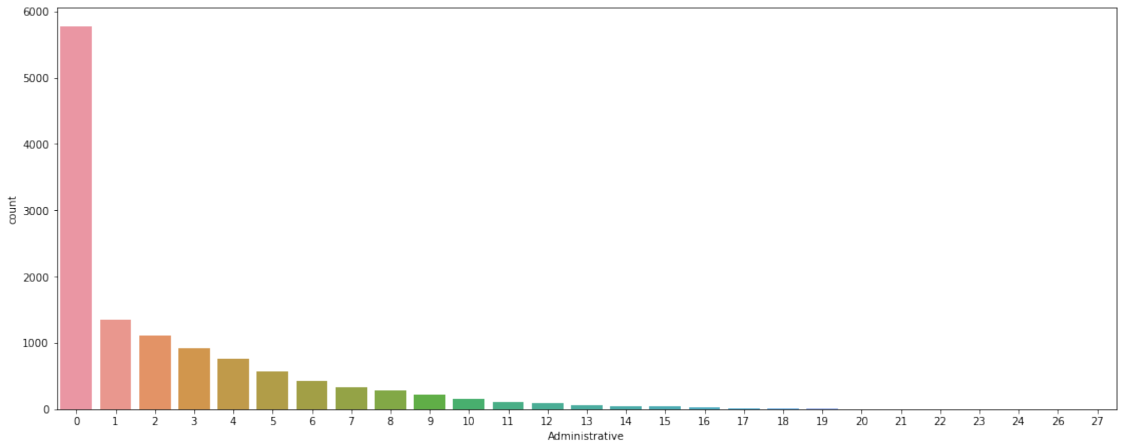
```
plt.savefig('output9.png', dpi=300, bbox_inches='tight')
plt.show()
```



From the above graphs we can observe that the distribution of Attributes related to Web Pages vs Revenue

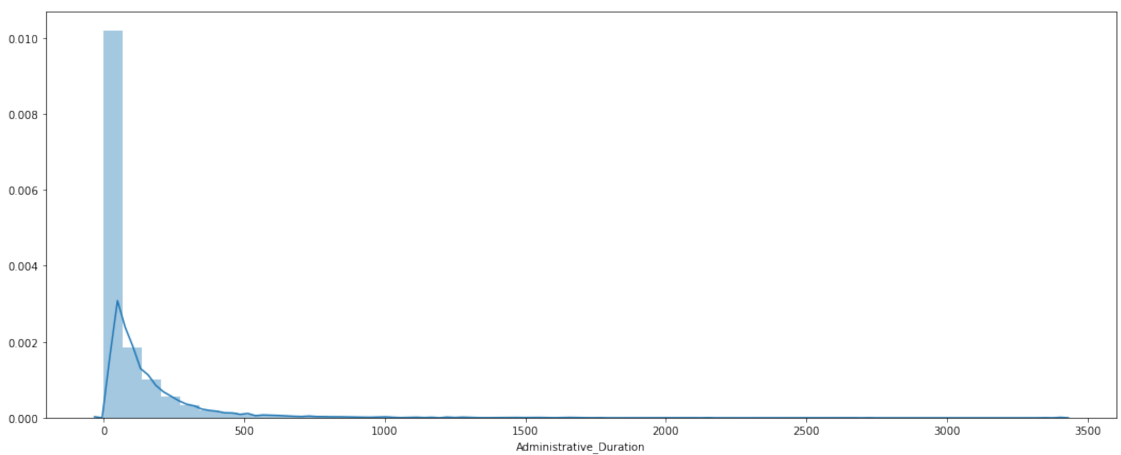
1.10 Number of Visits to "Administrative" type Pages

[27]:
```
plt.figure(figsize = (18,7))
sns.countplot(df['Administrative'])
plt.show()
```



1.11 Distribution of time spent by a user in Administrative Pages

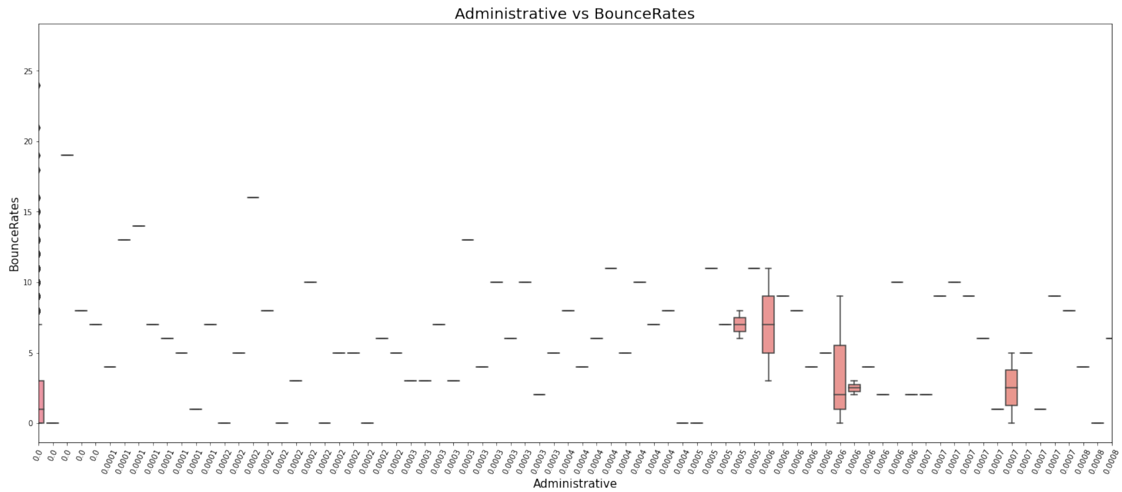
[28]:
```
plt.figure(figsize = (18,7))
sns.distplot(df['Administrative_Duration'])
plt.show()
```



1.12 Bounce Rates in Administrative Pages

[29]:
```
plt.figure(figsize = (25,10))
axa=sns.boxplot(df['BounceRates'], df['Administrative'])
plt.title('Administrative vs BounceRates', fontsize = 20)
plt.xlabel('Administrative', fontsize = 15)
plt.ylabel('BounceRates', fontsize = 15)
```

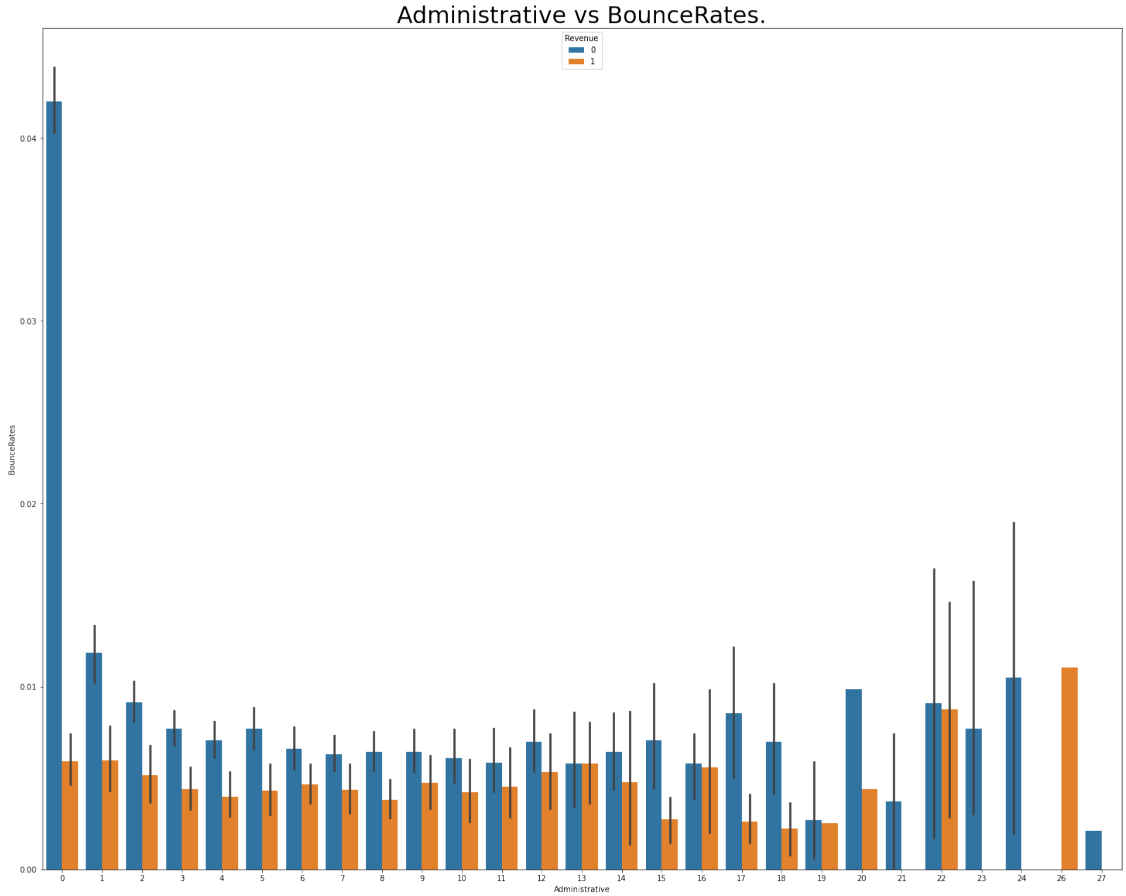
```
labels = [item.get_text() for item in axa
          .get_xticklabels()]
axa.set_xticklabels([str(round(float(label), 4)) for label in labels])
plt.xticks(rotation=65)
plt.xlim(0,75)
plt.show()
```



1.13 Percentage of users who enter the website and exit it without triggering any additional tasks after Visiting Administrative Pages

[30]:
```
plt.figure(figsize = (25,20))
sns.barplot(x = df['Administrative'], y = df['BounceRates'], hue=df['Revenue'])
plt.title('Administrative vs BounceRates.', fontsize = 30)
```

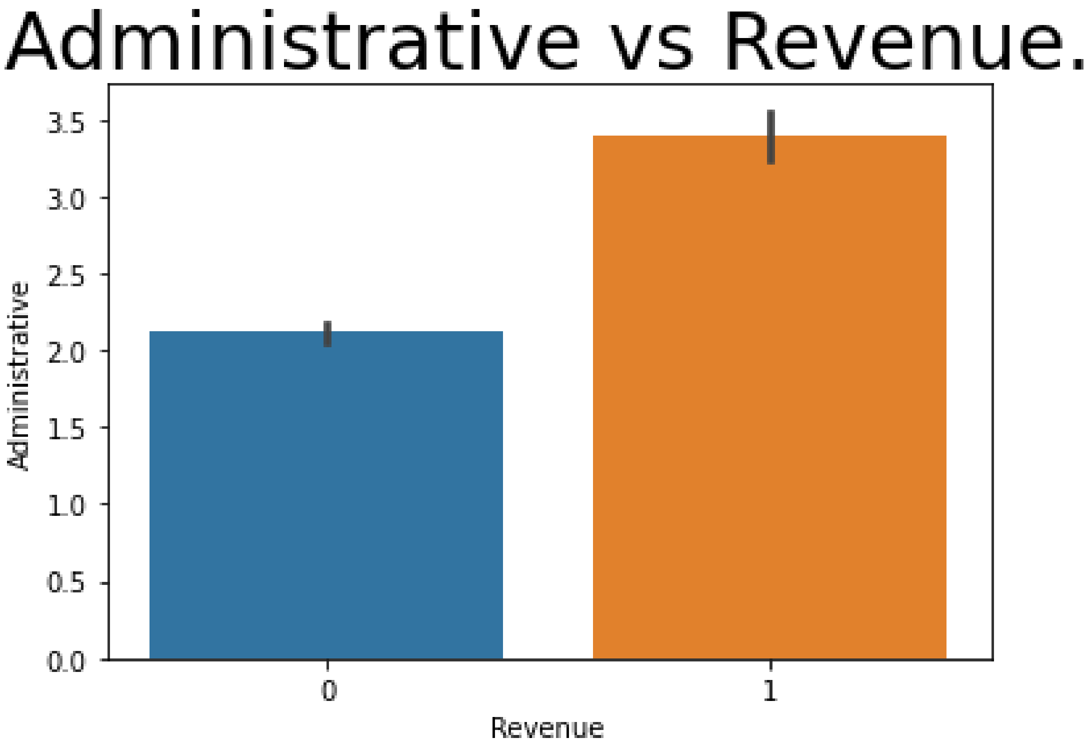
[30]:
```
Text(0.5, 1.0, 'Administrative vs BounceRates.')
```



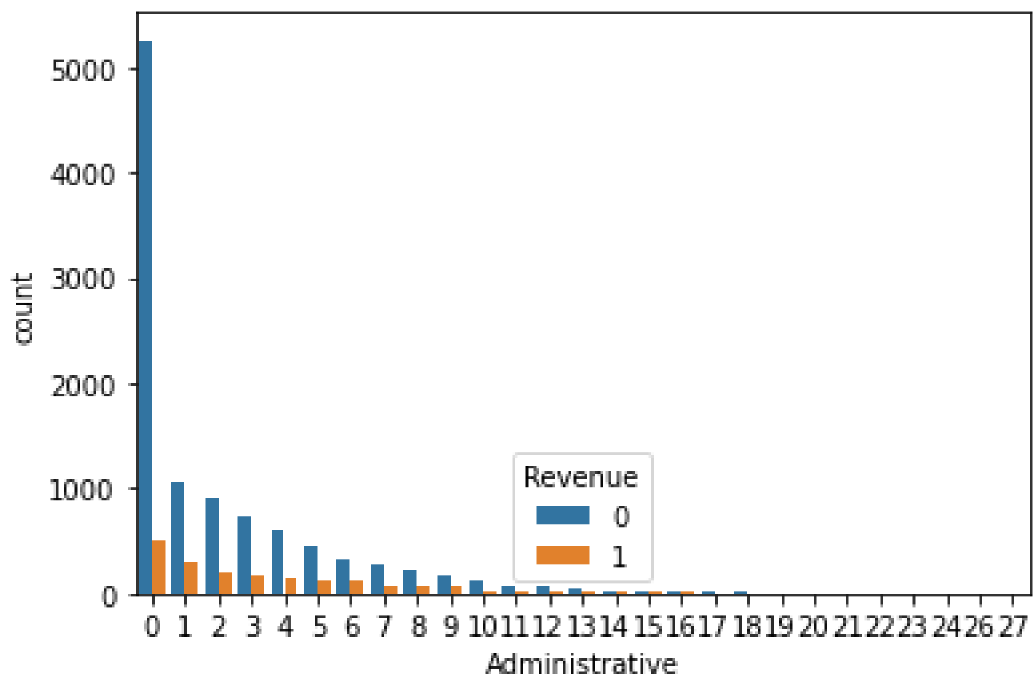
1.14 Revenue measured against visits to Administrative Pages

[31]:
```
sns.barplot(x = df['Revenue'], y = df['Administrative'])
plt.title('Administrative vs Revenue.', fontsize = 30)
```

[31]: Text(0.5, 1.0, 'Administrative vs Revenue.')



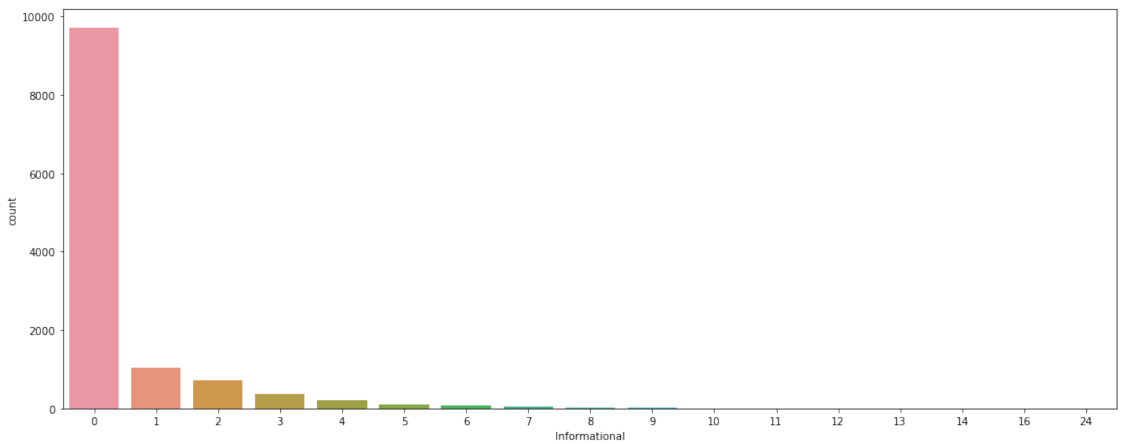
[32]:
```
sns.countplot(x=df['Administrative'], hue=df['Revenue'], data=pd.melt(df))
plt.show()
```



From the above graphs we can observe that the behavior of a user after visting an "Administrative" page on the website, and how it relates to the Class label Revenue and the attribute "BounceRates"

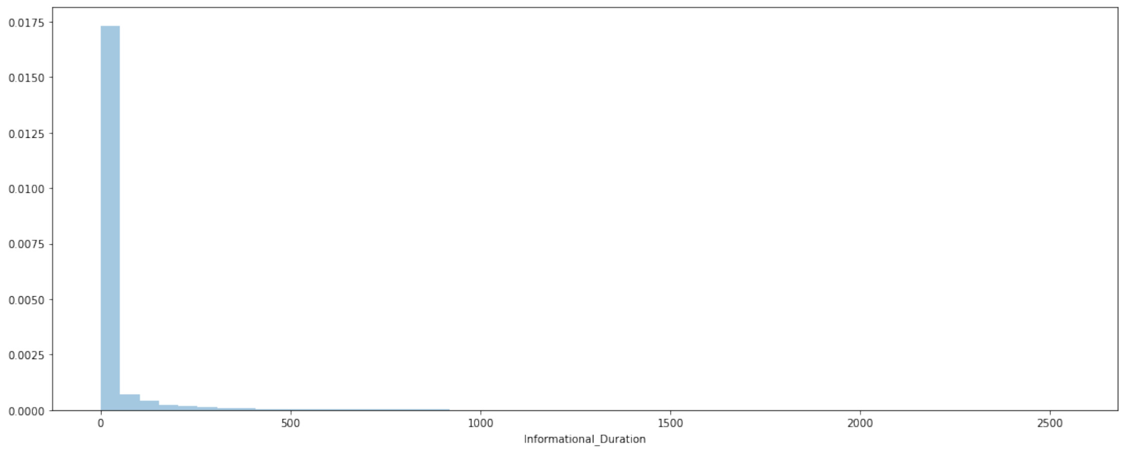
1.15 Number of Visits to "Informational" type Pages

[33]:
```
plt.figure(figsize = (18,7))
sns.countplot(df['Informational'])
plt.show()
```



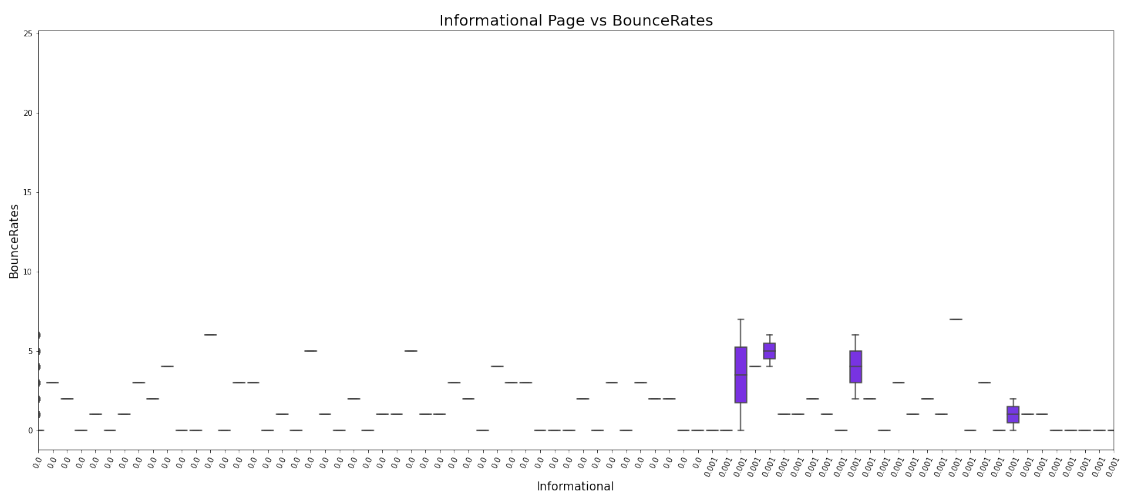
1.16 Distribution of time spent by a user in Informational Pages

[34]:
```
plt.figure(figsize = (18,7))
sns.distplot(df['Informational_Duration'])
plt.show()
```



1.17 Bounce Rates in Informational Pages

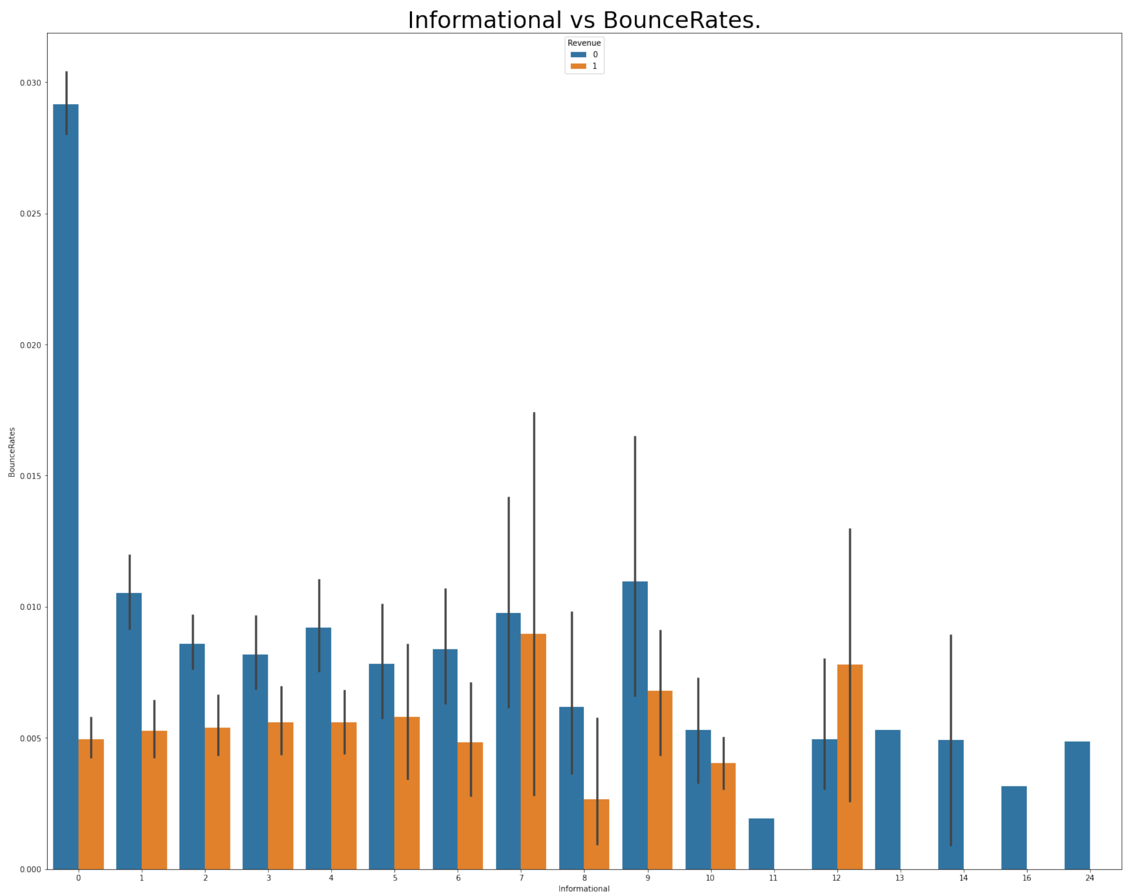
[35]:
```python
plt.figure(figsize = (25,10))
axd=sns.boxplot(df['BounceRates'],df['Informational'], palette = 'rainbow')
plt.title('Informational Page vs BounceRates', fontsize = 20)
plt.xlabel('Informational', fontsize = 15)
plt.ylabel('BounceRates', fontsize = 15)
labels = [item.get_text() for item in axd
          .get_xticklabels()]
axd.set_xticklabels([str(round(float(label), 3)) for label in labels])
plt.xticks(rotation=65)
plt.xlim(0,75)
plt.show()
```



1.18 Percentage of users who enter the website and exit it without triggering any additional tasks after Visiting Administrative Pages

[36]:
```
plt.figure(figsize = (25,20))
sns.barplot(x = df['Informational'], y = df['BounceRates'], hue=df['Revenue'])
plt.title('Informational vs BounceRates.', fontsize = 30)
```

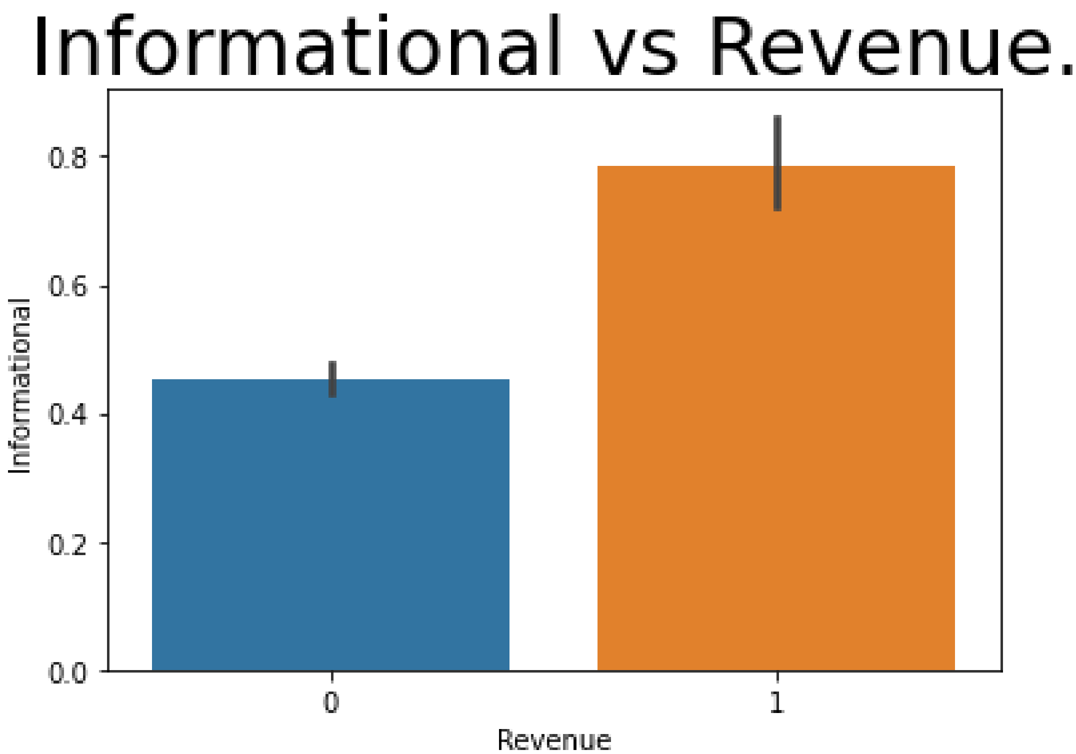
[36]: Text(0.5, 1.0, 'Informational vs BounceRates.')



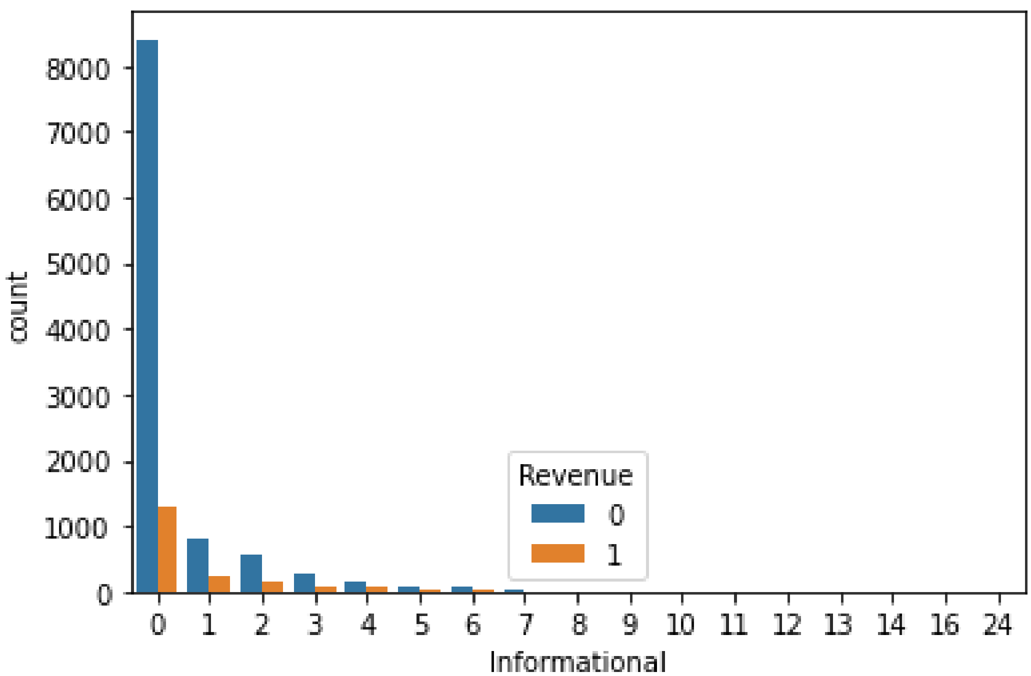
1.19 Revenue measured against visits to Informational Pages

[37]:
```
sns.barplot(x = df['Revenue'], y = df['Informational'])
plt.title('Informational vs Revenue.', fontsize = 30)
```

[37]: Text(0.5, 1.0, 'Informational vs Revenue.')



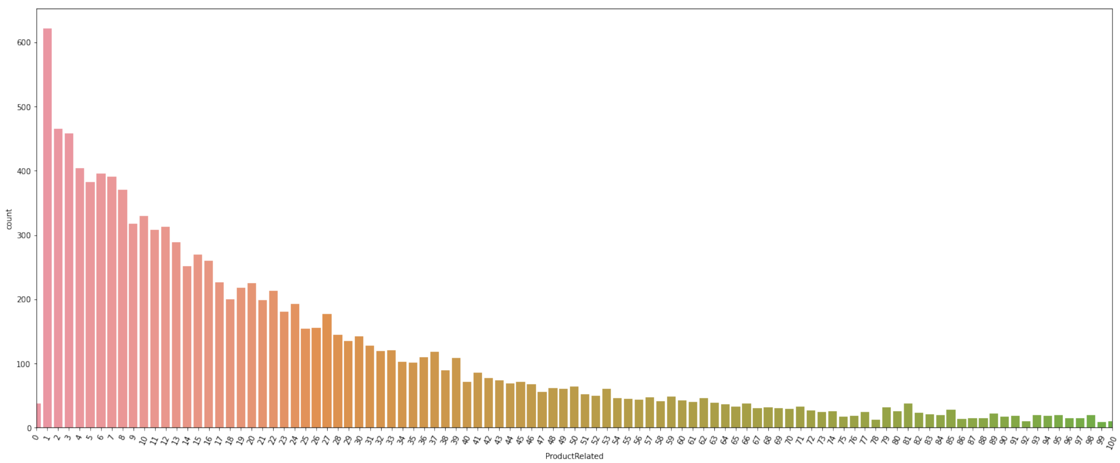
[38]:
```
sns.countplot(x=df['Informational'], hue=df['Revenue'], data=pd.melt(df))
plt.show()
```



From the above graphs we can observe that the behavior of a user after visting an "Informational" page on the website, and how it relates to the Class label Revenue and the attribute "BounceRates"

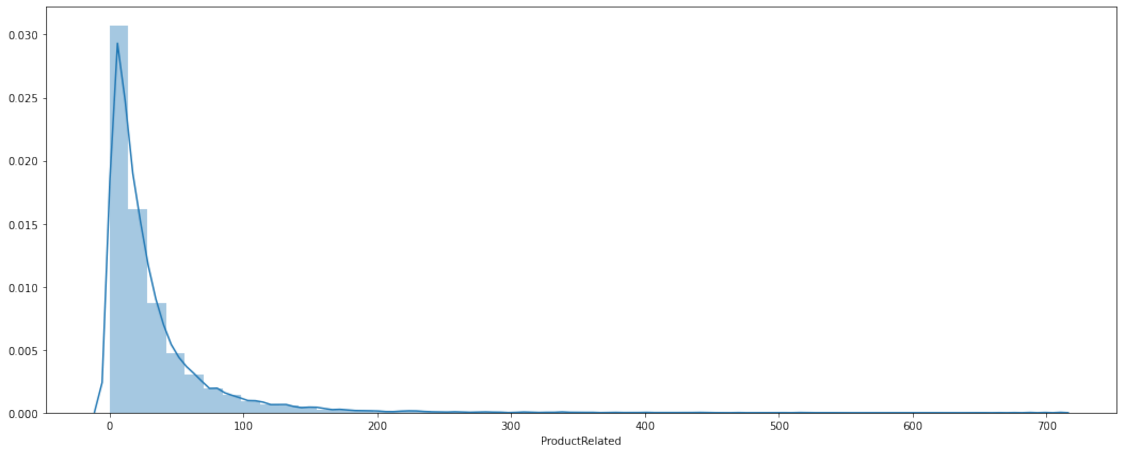
1.20 Number of Visits to "Product Related" Pages

[39]:
```
plt.figure(figsize = (25,10))
sns.countplot(df['ProductRelated'])
plt.xlim(0,100)
plt.xticks(rotation=65)
plt.show()
```



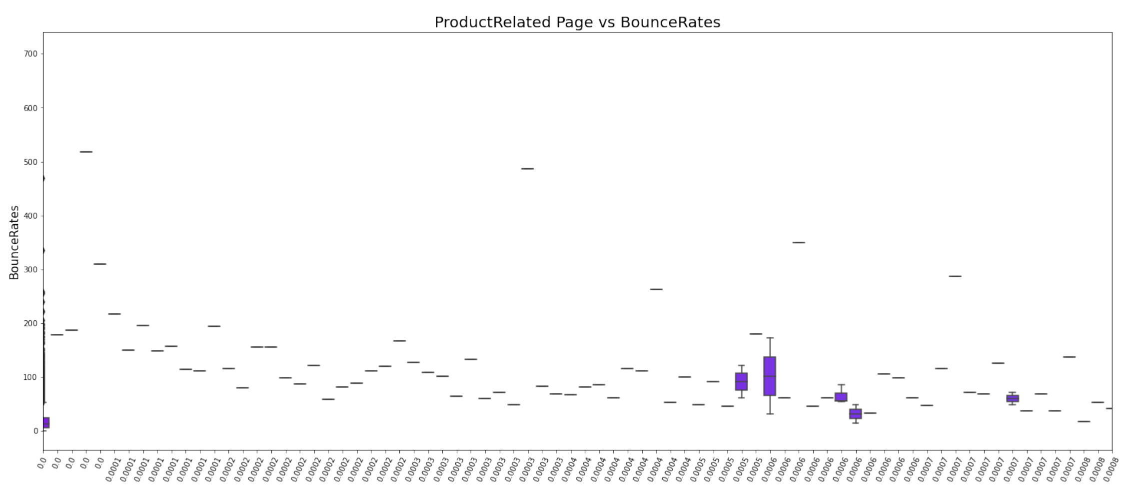
1.21 Distribution of time spent by a user in Product Related Pages

[40]:
```
plt.figure(figsize = (18,7))
sns.distplot(df['ProductRelated'])
plt.show()
```



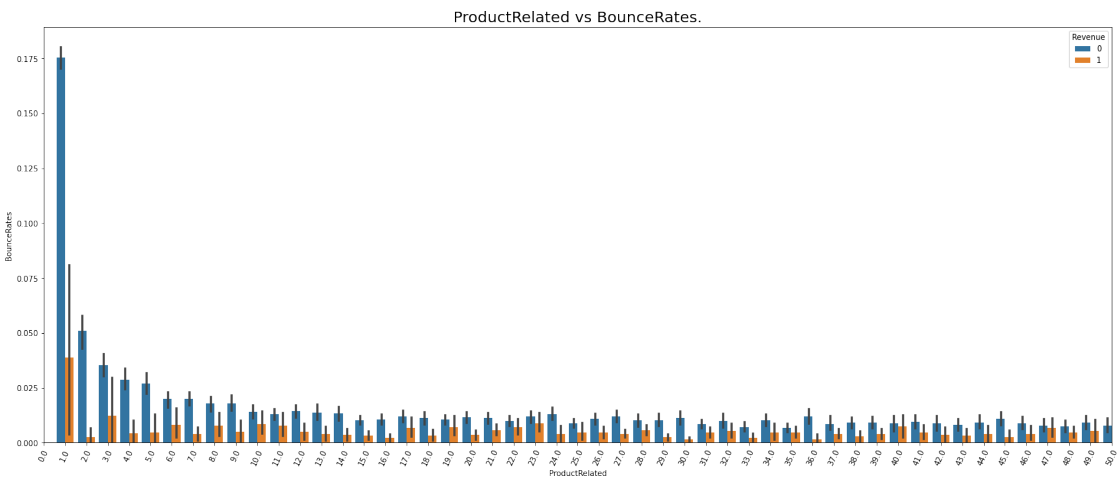
1.22 Bounce Rates in ProductRelated Pages

[41]:
```
plt.figure(figsize = (25,10))
axp= sns.boxplot(df['BounceRates'], df['ProductRelated'], palette = 'rainbow')
plt.title('ProductRelated Page vs BounceRates', fontsize = 20)
plt.xlabel('ProductRelated', fontsize = 15)
plt.ylabel('BounceRates', fontsize = 15)


labels = [item.get_text() for item in axp.get_xticklabels()]
axp.set_xticklabels([str(round(float(label), 4)) for label in labels])

plt.xticks(rotation=65)
plt.xlim(0,75)
plt.show()
```



1.23 Percentage of users who enter the website and exit it without triggering any additional tasks after ProductRelated Pages

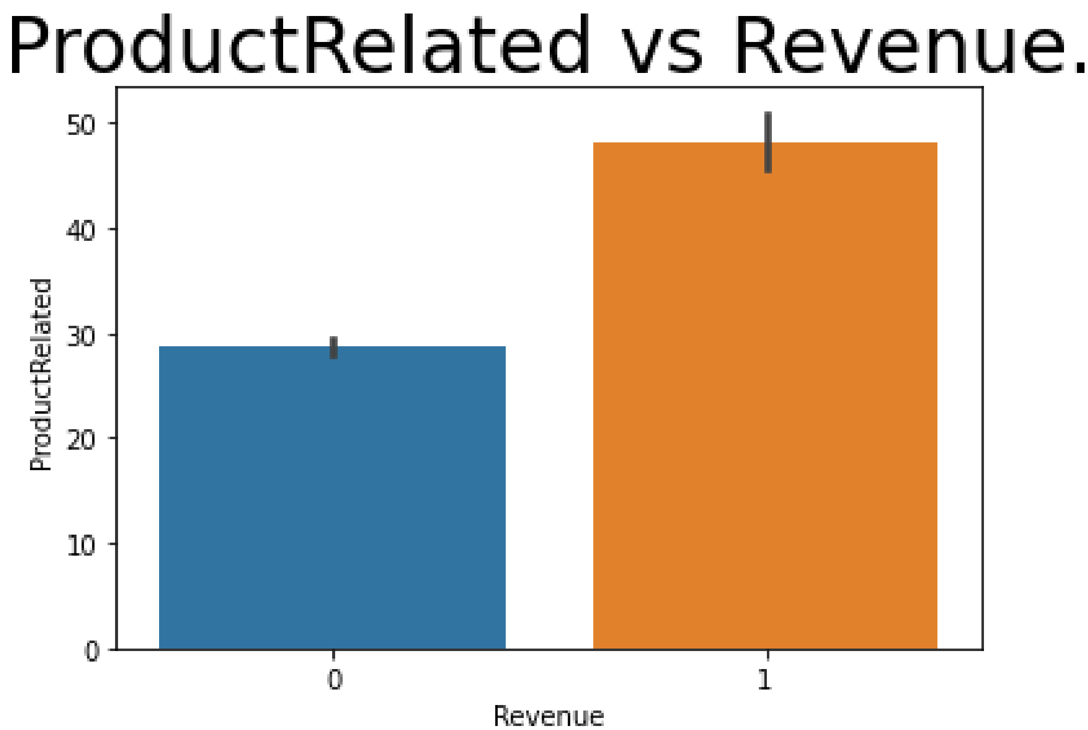
[42]:
```
plt.figure(figsize = (25,10))
axx=sns.barplot(x = df['ProductRelated'], y = df['BounceRates'], hue=df['Revenue'])
plt.title('ProductRelated vs BounceRates.', fontsize = 20)
labels = [item.get_text() for item in axx
          .get_xticklabels()]
axx.set_xticklabels([str(round(float(label), 4)) for label in labels])
plt.xticks(rotation=65)
plt.xlim(0,50)
plt.show()
```



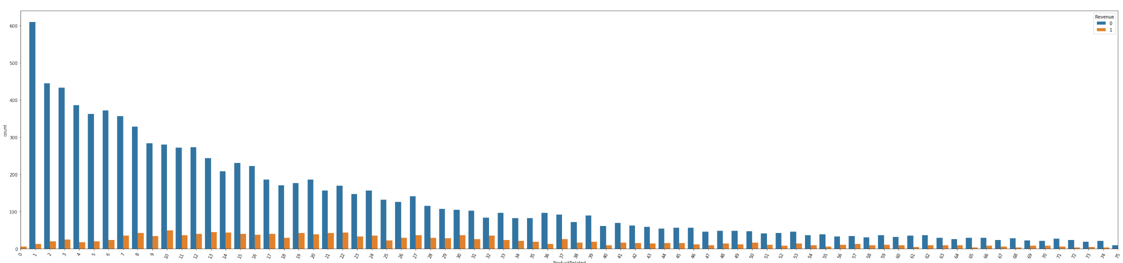
1.24 Revenue measured against visits to Product Related Pages

[43]:
```
sns.barplot(x = df['Revenue'], y = df['ProductRelated'])
plt.title('ProductRelated vs Revenue.', fontsize = 30)
```

[43]: Text(0.5, 1.0, 'ProductRelated vs Revenue.')



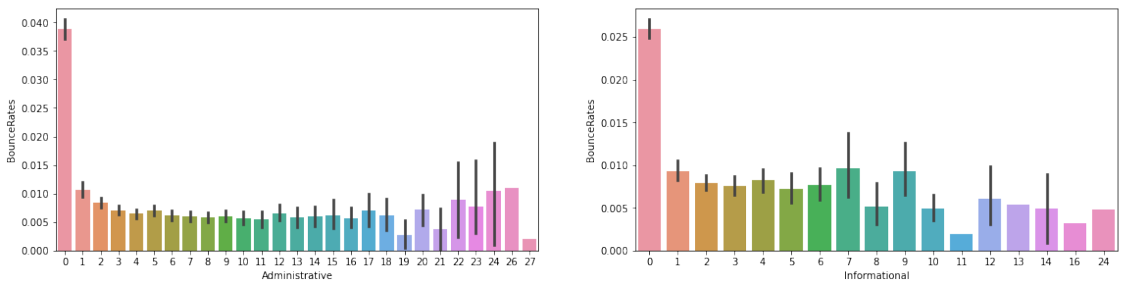
[44]:
```
plt.figure(figsize=(45, 10))
sns.countplot(x=df['ProductRelated'], hue=df['Revenue'], data=pd.melt(df))
plt.xlim(0,75)
plt.xticks(rotation=65)
plt.show()
```



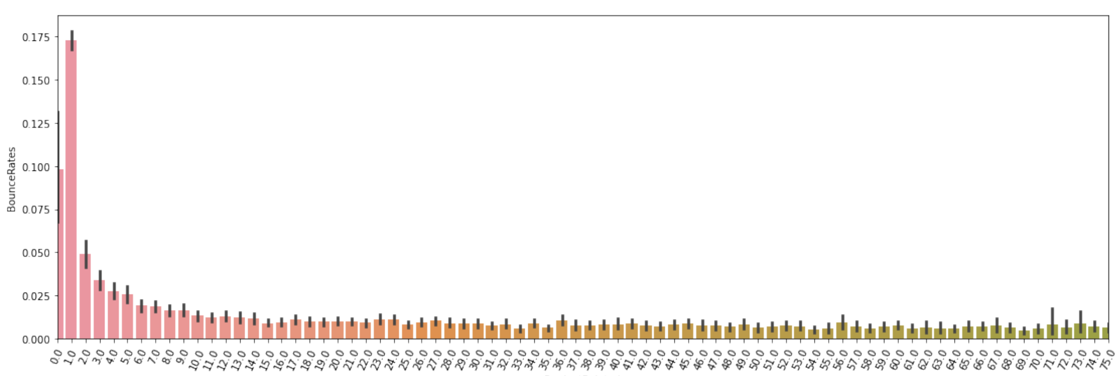
From the above graphs we can observe that the behavior of a user after visting a "Product Related" page on the website, and how it relates to the Class label Revenue and the attribute "BounceRates"

[45]:
```
fig = plt.figure(figsize=(30, 10))

x2 = fig.add_subplot(2, 3, 2)
```

```
x3 = fig.add_subplot(2, 3, 3)
sns.barplot(x = df['Administrative'], y = df['BounceRates'],ax=x2)
sns.barplot(x = df['Informational'], y=df['BounceRates'], ax=x3)
plt.savefig('output10.png', dpi=300, bbox_inches='tight')
plt.show()
```



[46]:
```
fig = plt.figure(figsize=(45, 10))

fig.add_subplot(2, 3, 1)

plot1=sns.barplot(x = df['ProductRelated'], y = df['BounceRates'])

labels = [item.get_text() for item in axx
          .get_xticklabels()]
plot1.set_xticklabels([str(round(float(label), 4)) for label in labels])
plt.xticks(rotation=65)
plt.xlim(0,75)
plt.tight_layout()
plt.savefig('output11.png', dpi=300, bbox_inches='tight')
plt.show()
```



Based on the above comparisons, it can be observed, that "Product Relates type" pages are most likely to have a high Bounce rate, meaning the customer

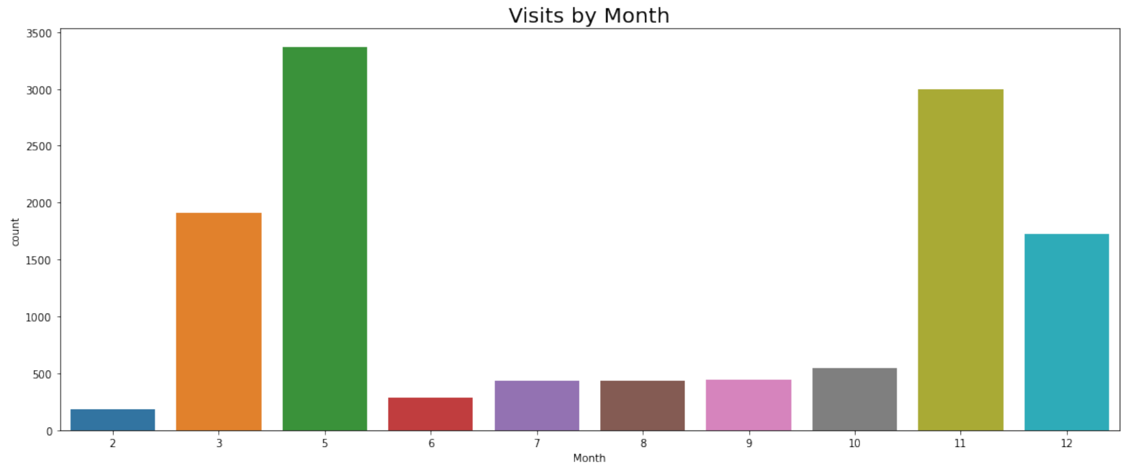
is most likely to not proceed deeper into the website and by doing so, not triggering aditional tasks, this can be understood since it has the more visits that any other type of Pages. The value of "Bounce Rate" feature for a web page refers to the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session.

1.25 Month sessions analysis

[47]:
```
print(df['Month'].value_counts())
```

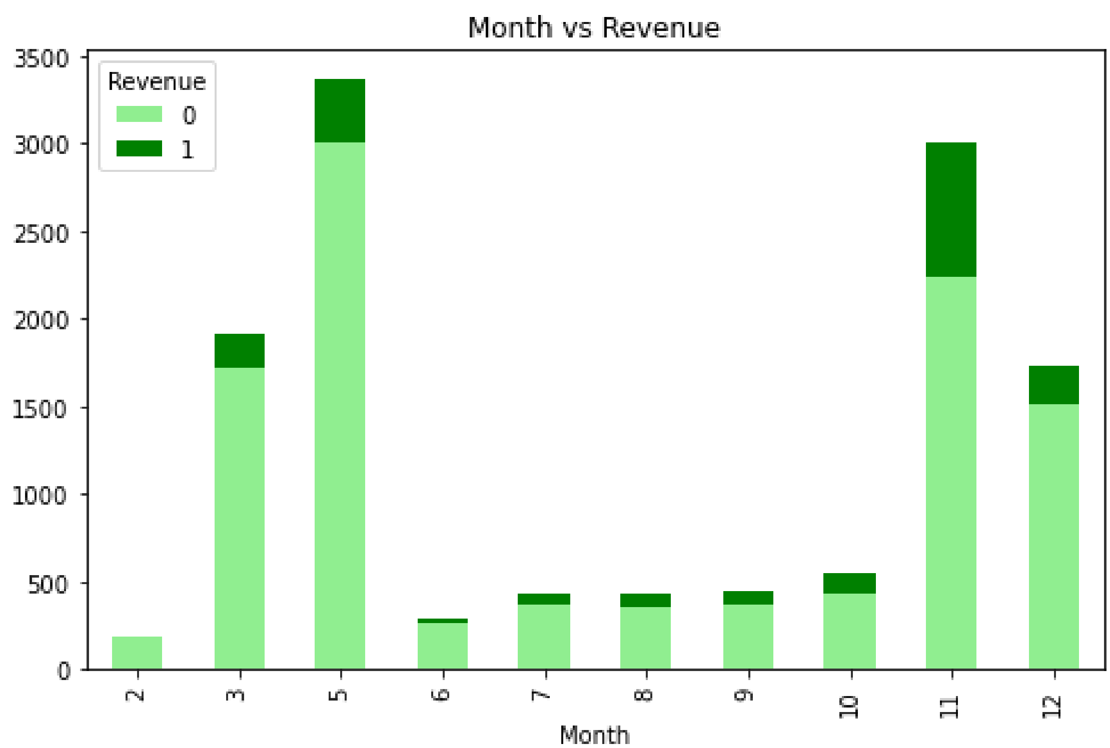
```
5     3364
11    2998
3     1907
12    1727
10     549
9      448
8      433
7      432
6      288
2      184
Name: Month, dtype: int64
```

[48]:
```

plt.figure(figsize = (18,7))
sns.countplot(df['Month'])
plt.title('Visits by Month', fontsize = 20)
plt.show()
```



1.26 Revenue by Month

[49]:
```
data = pd.crosstab(df['Month'], df['Revenue'])
data.plot(kind = 'bar', stacked = True, figsize = (8, 5), color =
 ↪['lightgreen', 'green'])
plt.title('Month vs Revenue')
plt.savefig('output12.png', dpi=300, bbox_inches='tight')
plt.show()
```



Based on the above graphs we can observe the following: **- The behavior of sessions by Month, and how it relates to the class label Revenue -We can note how the months: March, May, November and december have the higher number of visits - We can also observe how November has the highest revenue, we can infer that this is because people buy in advance of the Holidays**

1.27 Visitor Type analysis

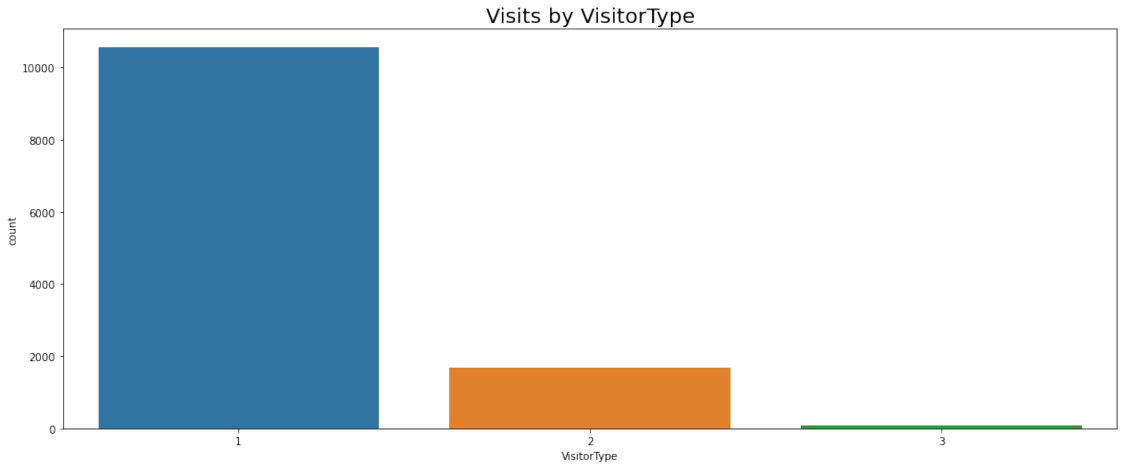
[50]:
```
print(df['VisitorType'].value_counts())
```

```
1    10551
2     1694
```

```
3       85
Name: VisitorType, dtype: int64
```

[51]:

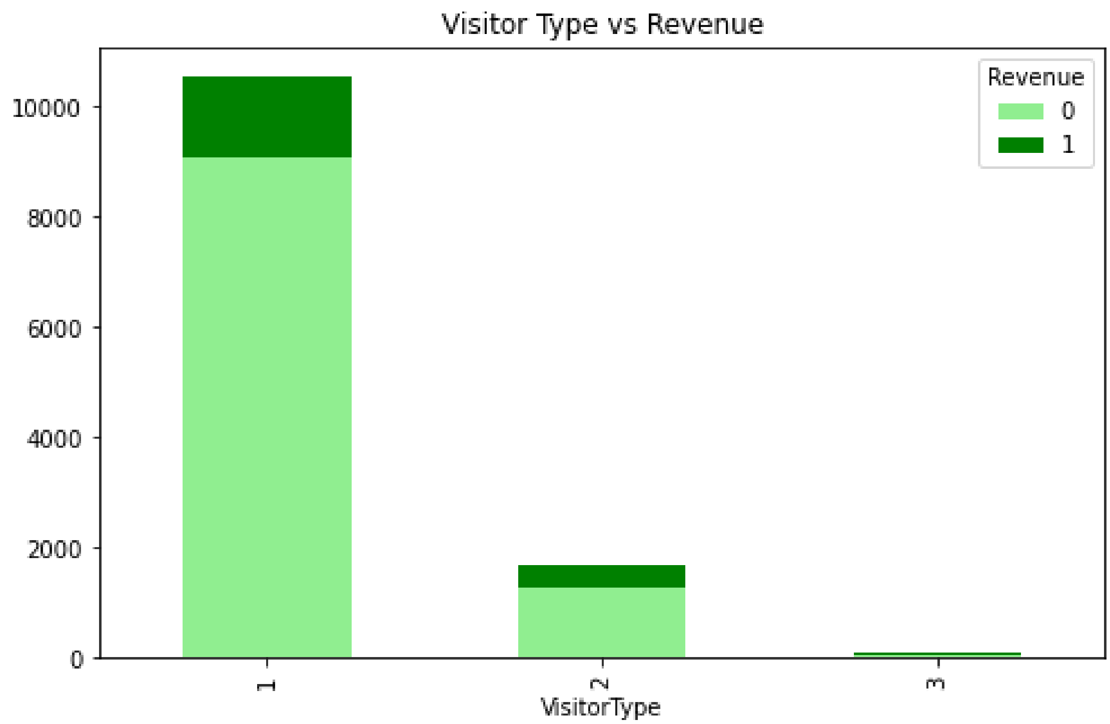
```
plt.figure(figsize = (18,7))
sns.countplot(df['VisitorType'])
plt.title('Visits by VisitorType', fontsize = 20)
plt.show()
```



1.28 Visitor Type by Revenue

[52]:

```
data = pd.crosstab(df['VisitorType'], df['Revenue'])
data.plot(kind = 'bar', stacked = True, figsize = (8, 5), color =␣
 ↪['lightgreen', 'green'])
plt.title('Visitor Type vs Revenue')
plt.savefig('output13.png', dpi=300, bbox_inches='tight')
plt.show()
```



Based on the above graphs we can observe the following: Visitor type as "New Visitor," "Returning Visitor," and "Other" - **The number of visits by visitor Type and how it relates to the class label Revenue - We can note how the Visitor Type 1 "New Visitor" has the higher number of visits - We can also observe how Visitor Type 1 has the highest revenue**

1.29 Save dataset with modified values to new cvs file

[53]:
```
#df.to_csv('Onlineshoppersdata(1).csv', index=False)
```

2 Feature Selection using SULOV (Searching for Uncorrelated List of Variables)

[54]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from featurewiz import featurewiz
np.random.seed(0)
```

[55]:
```python
x = df.drop(['Revenue'],axis=1)
y = df.Revenue.values
x_scaled =  StandardScaler().fit_transform(x)
X_train, x_valid, y_train, y_valid = train_test_split(x_scaled,y,test_size = 0.
↪2,stratify=y, random_state=42)
classifier = RandomForestClassifier()
classifier.fit(X_train,y_train)
```

[55]:
```
RandomForestClassifier()
```

[56]:
```python
# make prediction
preds = classifier.predict(x_valid)
# check performance

ac= (accuracy_score(preds,y_valid)*100)

print('The accuracy is ' + str(round(ac,2)))
```

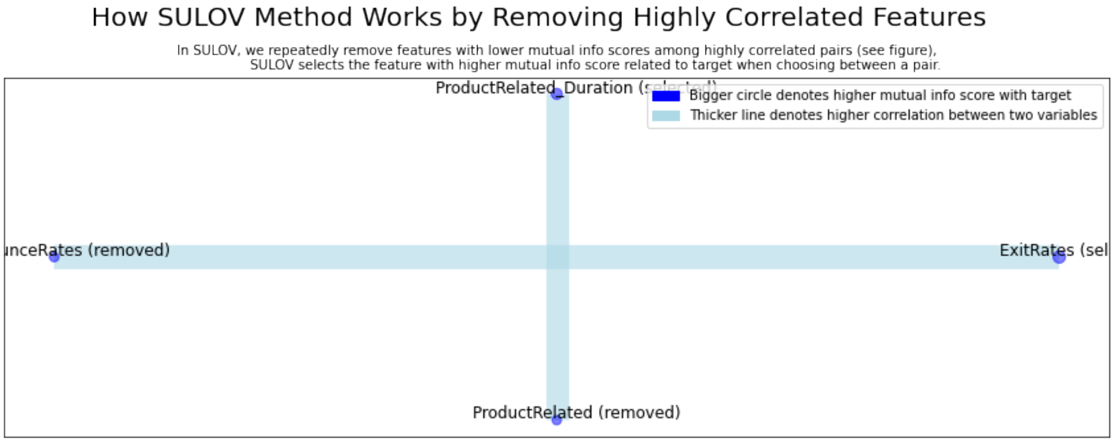
```
The accuracy is 90.31
```

Automatic feature selection by using featurewiz package

[57]:
```python
target = 'Revenue'
features, train = featurewiz(df, target, corr_limit=0.7, verbose=2, sep=",",
header=0,test_data="", feature_engg="", category_encoders="")
```

```
Skipping feature engineering since no feature_engg input…
Skipping category encoding since no category encoders specified in input…
Loading train data…
Shape of your Data Set loaded: (12330, 18)
Loading test data…
    Filename is an empty string or file not able to be loaded
############## C L A S S I F Y I N G  V A R I A B L E S  ####################
Classifying variables in data set…
    17 Predictors classified…
        No variables removed since no ID or low-information variables found in
data set
No GPU active on this device
    Running XGBoost using CPU parameters
Removing 0 columns from further processing since ID or low information variables
    columns removed: []
    After removing redundant variables from further processing, features left =
17
#### Single_Label Binary_Classification Feature Selection Started ####
Searching for highly correlated variables from 17 variables using SULOV method
#####  SULOV : Searching for Uncorrelated List Of Variables (takes time…)
############
There are no null values in dataset.
```

```
    Removing (2) highly correlated variables:
    ['ProductRelated', 'BounceRates']
    Following (15) vars selected: ['Administrative', 'Administrative_Duration',
'Informational', 'Informational_Duration', 'PageValues', 'SpecialDay', 'Month',
'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
'Weekend', 'ExitRates', 'ProductRelated_Duration']
```




```
    Adding 0 categorical variables to reduced numeric variables  of 15
############## F E A T U R E   S E L E C T I O N  ####################
Current number of predictors = 15
    Finding Important Features using Boosted Trees algorithm…
        using 15 variables…
        using 12 variables…
        using 9 variables…
        using 6 variables…
        using 3 variables…
Selected 15 important features from your dataset
    Time taken (in seconds) = 308
Returning list of 15 important features and dataframe.
```

[58]:
```
print(features)
```

```
['PageValues', 'Month', 'VisitorType', 'ExitRates', 'ProductRelated_Duration',
'Administrative', 'Administrative_Duration', 'SpecialDay',
'Informational_Duration', 'Informational', 'TrafficType', 'Weekend',
'OperatingSystems', 'Region', 'Browser']
```

[59]:
```
#split data into feature and target
X_new = train.drop(['Revenue'],axis=1)
y = train.Revenue.values
```

[60]:
```
# preprocessing the features
X_scaled =  StandardScaler().fit_transform(X_new)
```

[61]:
```
X_train, X_valid, y_train, y_valid = train_test_split(X_scaled,y,test_size = 0.
↪2,stratify=y, random_state=42)
# create and train classifier
classifier = RandomForestClassifier()

classifier.fit(X_train,y_train)
```

[61]:
```
RandomForestClassifier()
```

[62]:
```
# make prediction
preds = classifier.predict(X_valid)
# check performance
ac1= accuracy_score(preds,y_valid)*100
print('The accuracy of the model with 15 features is ' + str(round(ac1,2)))
```

```
The accuracy of the model with 15 features is 89.46
```

The model is slightly less accurate when we remove ['ProductRelated', 'BounceRates'] from the list of attributes

2.1 Feature selection using Feature Importance

[63]:
```
# Feature Importance with Extra Trees Classifier
from pandas import read_csv
from sklearn.ensemble import ExtraTreesClassifier

X = df.drop(['Revenue'],axis=1)

Y = df.Revenue.values

# feature extraction
model = ExtraTreesClassifier(n_estimators=10)
model.fit(X, Y)

importance = model.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
        print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```

```
Feature: 0, Score: 0.05226
Feature: 1, Score: 0.04672
```

```
Feature: 2, Score: 0.03031
Feature: 3, Score: 0.02754
Feature: 4, Score: 0.06286
Feature: 5, Score: 0.06243
Feature: 6, Score: 0.05138
Feature: 7, Score: 0.08764
Feature: 8, Score: 0.33959
Feature: 9, Score: 0.00713
Feature: 10, Score: 0.05329
Feature: 11, Score: 0.03190
Feature: 12, Score: 0.03203
Feature: 13, Score: 0.04361
Feature: 14, Score: 0.04254
Feature: 15, Score: 0.01463
Feature: 16, Score: 0.01414
```

```
        ␣
 ↪---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-63-302df9005d3c> in <module>
         16         print('Feature: %0d, Score: %.5f' % (i,v))
         17 # plot feature importance
    ---> 18 pyplot.bar([x for x in range(len(importance))], importance)
         19 pyplot.show()


        NameError: name 'pyplot' is not defined
```

Bar chart to show the feature importance scores for the attributes

2.2 Feature selection using Univariate Selection

[ ]:
```
#apply SelectKBest class to extract top 10 best features
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from matplotlib import pyplot


X = df.drop(['Revenue'],axis=1)
Y = df.Revenue.values

bestfeatures = SelectKBest(score_func=chi2, k=17)
```

```
fit = bestfeatures.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(17,'Score'))  #print  features with score
```

Univariate feature selection examines each attribute individually to determine the strength of the relationship of it feature with the Class label "Revenue".

2.2.1 After running 3 different feature selection methods, and the three of them ranking the attributes differently. I have decided to not drop any of the attributes to run the Classifiers.