Predictive Modeling via Logistic Regression for Online Purchasing Intention

Alexa R. Fisher

Western Governors University

Degree: M.S. Data Analytics

Table of Contents

Part I. Research Question for Data Analysis

A. Research Question for Analysis

The research question for this capstone project was "What attributes within the captured Google Analytics metrics can contribute to the possibility of a customer purchasing an item?".

In current society, online purchasing became a common occurrence. The utilization of Google Analytics within e-commerce offered a wide variety of insights to improve profitability (Kumar, 2023). It was captured by user behavior within the website platform such as the number of times each page was visited, how long a user interacted with the page, or even if a product was purchased. The data was limited to one year to ensure that all the annual holidays were represented.

The usage of logistic regression could identify if any of these metrics provided a significant relationship between whether or not an item was purchased. This justified the research analysis as the results could ascertain which metrics could be used for strategic business marketing.

For this data analysis, the following null and alternative hypotheses were established:

Null hypothesis: The logistic regression model shows that the predictor variable of "Revenue" and the explanatory variables do not have a statistically significant relationship.

Alternate Hypothesis: The logistic regression model shows a statistically significant relationship between the predictor variable "Revenue" and explanatory variables.

To reject the null hypothesis, the optimized logistic regression model accuracy would be greater than 80% and statistically significant attributes identified. Statistical significance was noted as a p-value of less than 0.05. To fail to reject the hypothesis, the model would have a p-value of greater than 0.05, deeming there was not a statistically significant relationship between the target and explanatory variables.

Part II. Data Collection

B. Summary of Data Collection Process

The dataset utilized for this project was obtained from the UC Irvine Machine Learning Repository via the UC Irvine website (Sakar & Kastro, 2018). The UCI Machine Learning Repository contained a variety of datasets published and released to the public for adaptation. The Online Shoppers dataset was limited to a one-year period. The data consisted of 12,330 instances and 18 variables. The variables consisted of 10 quantitative variables and 8 qualitative variables. The below provided a brief description of each of the variables.

- Administrative: The number of Administrative type pages a user visited.
- Administrative_Duration: The time a user spends on an Administrative page.
- Informational: The number of Informational type pages a user visited.
- Informational_Duration: The time a user spends on an Information page.
- ProductRelated: The number of Product Related type pages a user visited.

- ProductRelated_Duration: The amount of time a user spent on a Product related page.
- BounceRates: The percentage of visitors who enter the website and exit without triggering any other tasks.
- ExitRates: The percentage of page views on the website ending on a specific page.
- PageValues: The average value of the page that a user visited before completing a transaction.
- Month: The month the page view occurred.
- SpecialDay: The value indicating the closeness of the browsing date to a special day or holiday.
- OperatingSystems: The integer value representing a user's operating system.
- Browser: The integer value representing the browser a user was using.
- Region: The integer value representing the region a user was located.
- TrafficType: The integer value representing what type of traffic the user is categorized into.
- VisitorType: Identification of whether a user is a new visitor, returning visitor, or other.
- Weekend: The Boolean value if the session was on a weekend.
- Revenue: The Boolean value of whether the user completed the purchase or not.

One of the main advantages of this data-gathering methodology was its simplicity. The data was downloadable in a comma-separated value (.csv) format. This format was easily imported into a dataframe with the use of the pandas' function .read_csv(). This function was noted as one of the most used functions within the pandas package as it allowed for external datasets to be imported and manipulated further for analysis.

One of the main disadvantages of this methodology was the time limitation of the dataset. As this dataset was only limited to a one-year period. With this limitation, it was not possible to ascertain if the observations were a typical year for the e-commerce website.

There were a few challenges during the data-gathering process. The dataset did not come with a clear data dictionary. Most of the definitions were based on the values noted during the exploration of the data. For example, the duration variables were float values, but it was impossible to tell if they were in seconds, minutes, or some other measurement. This challenge was overcome by analyzing the values noted within the dataset. With the use of the describe() function, the summary statistics were displayed to show the minimum and maximum values noted for the duration variables. For example, "Administrative_Duration" has a minimum value of "0" and a maximum value of "3398.75". As it was already noted as user behavior on the e-commerce website, the likelihood of maximum value being in hours was very slim. Even minutes seemed abnormal as the value would have equaled a conversion to 56 hours in one session. This concluded the measurement of seconds to be ideal as the total conversion to minutes was about 56 minutes. With a little bit of research, it was confirmed that Google Analytics session metrics were captured in seconds (Google, n.d.).

Part III. Data Extraction and Preparation

C. Data Extraction and Preparation Process

The data extraction and preparation process included various steps on the online purchasing intention dataset. There were three key elements to this process which included data cleaning, data exploration, and data wrangling. The procedures were applied prior to performing the data analysis. The tools included the use of the Python programming language, Jupyter Notebook, and a plethora of libraries such as Pandas, NumPy, Matplotlib, and Seaborn.

Python was a programming language that supported data analysis with its use of various packages and libraries. One of its advantages was its great readability and ease of use. One main disadvantage was Python was a high-level programming language which could lead to slow execution (Joy, n.d.).

The usage of Jupyter Notebook allowed for visualizations and computations to be performed in a web-based environment. One of the advantages of this tool was code could be run cell by cell along with their results (Sjursen, 2020). On the opposite, a disadvantage would be slowness in operation. Each cell must run before the next was processed. It added processing time. The following libraries were useful for the preparation stage:

- Pandas have a large set of commands and features that allow for datasets to be analyzed. This point was noted as one of its advantages. The dataset was imported into a dataframe via the Pandas' .read_csv() function, which allowed manipulations throughout the exploration and wrangling stages. One of the disadvantages of Pandas was its poor documentation. The use of the pandas'

documentation did not provide adequate information on its functions without additional information located elsewhere. For example, “pd.set_options”. The documentation noted what it did but did not explain what parameters could be used (NumFOCUS, Inc, 2023).

```
#import the online purchasing csv file to be used.
#view dataset to ensure proper loading.
df = pd.read_csv('onlinepurchasing.csv')
pd.set_option('display.max_columns', None)
df.head()
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues
0	0	0.0	0	0.0	1	0.000000	0.20	0.20	0.0
1	0	0.0	0	0.0	2	64.000000	0.00	0.10	0.0
2	0	0.0	0	0.0	1	0.000000	0.20	0.20	0.0
3	0	0.0	0	0.0	2	2.666667	0.05	0.14	0.0
4	0	0.0	0	0.0	10	627.500000	0.02	0.05	0.0

- NumPy allows for different mathematical computations and the ability to work with arrays within the data cleaning and wrangling stages of the analysis. This could be one of its advantages as it allowed for various operations to work with large datasets efficiently. This was especially utilized within the treatment of the outliers. The NumPy .where() function was used to cap outliers to their IQR lower and upper bounds. One of the disadvantages of NumPy was the lack of flexibility. Once an array was created the data type and size could not be changed.
- Matplotlib and Seaborn were both useful in providing visualizations. An advantage of both of these visualization tools was the wide range of plot types available. Within the data preparation, these were used to generate boxplots, distplots, and barplots to view the various distributions as well as multivariate statistics. There was a disadvantage with these tools. One particular was the default style limitations. This was especially shown in the color palettes, the

distplots only have a few colors available, which made it hard when plotting multiple variables in assorted colors.

Before any data cleaning was completed, some initial data exploration was done. The shape method was used to show the number of entries and attributes of the dataframe.

```
df.shape
```

```
(12330, 18)
```

This was followed by the .info() function to show information about the variables such as the variable name, non-null value count, and data type. This dataset had a total of 18 variables, all of the variables had 12,330 non-null values. The datatypes varied between integers, floats, objects, and Boolean.

```
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
```

Once the brief initial data exploration was completed, the data cleaning process was initiated. The first step included the removal of duplicates, null values, and outliers. The .duplicated() function checked the full dataset for any duplicated values. In the case of this dataset, there were ~ 125 duplicates found. The duplicates were removed via the .drop_duplicates() function with the parameter *keep='last'*, which was used to keep the last occurrence. The shape method was used to confirm the remaining number of entries.

```
#finding duplicates
df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
12325    False
12326    False
12327    False
12328    False
12329    False
Length: 12330, dtype: bool
```

```
df.drop_duplicates(keep='last', inplace=True)
```

```
df.shape
```

```
(12205, 18)
```

The .isnull() function found any missing values within the dataset. This function was used in conjunction with the .sum() function to provide a total count of null values per variable. In the case of the online purchasing dataset, there were no null values found.

```
#finding nulls
df.isnull().sum()
```

```
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
```

Lastly, the detection and treatment of outliers were completed. The detection of outliers was computed via a created function, which was applied against the quantitative variables. The created function supplied the total amount of outliers as well as the minimum and maximum outlier values for each of those variables. The function utilized the interquartile range or IQR method. This method calculated the IQR by subtracting the first quantile from the third quantile. Within the function, there were two equations to mathematically calculate the lower and upper bounds for each variable. The calculations were as follows: Q1 – 1.5 * IQR = lower bound value and Q3 + 1.5 * IQR = upper bound value. In this method, the values found outside the two bounds were considered outliers. The utilization of box plots was used to show the distributions and graphically visualize the outliers. Once the outliers were identified, treatment was applied. The outliers were treated via another created function that used the same IQR method detection to find the boundaries. After the outlier boundaries were identified, the NumPy .where() function manipulated the values found outside of the lower and upper limits. This process was called

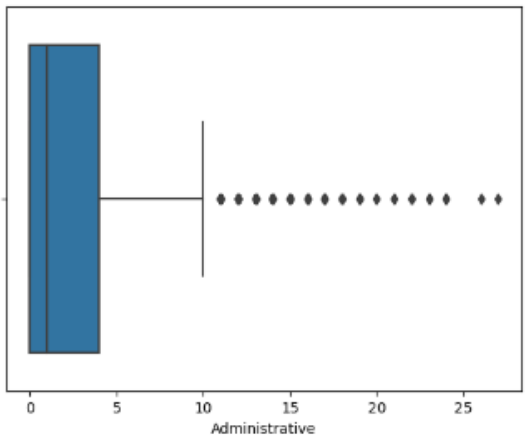
capping. Capping set the outliers to the closest lower or upper limit accordingly. After the outliers were capped, the box plots were re-plotted to ensure the outliers were resolved and the distributions did not change. The variables of “Informational”, “Informational_Duration”, “PageValues”, and “SpecialDay” ended up as zero values once cleaned. This caused some hesitation with the removal of these outliers as it removed all insights from those variables. For this analysis, a decision was made to keep these variables without treatment. This was justified, as it allowed for the variables to provide insight into the target variable.

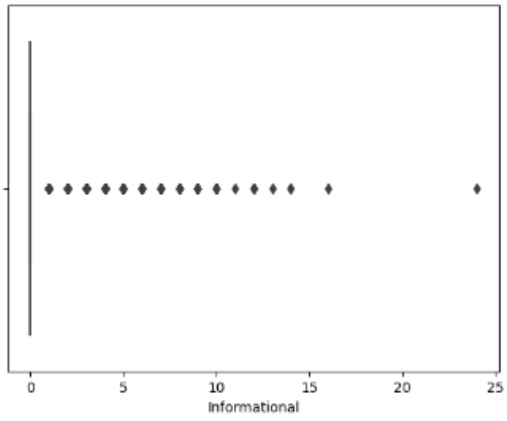
```
#finding outliers
def find_outliers(df, var):
    q1 = df[var].quantile(0.25)
    q3 = df[var].quantile(0.75)
    IQR = q3 - q1
    lowerbound = q1-(1.5*IQR)
    upperbound = q3+(1.5*IQR)
    outliers = df[var][((df[var] < (lowerbound)) | (df[var] > (upperbound)))]
    return outliers

#running created function on quantitative variables

outliers = find_outliers(df, 'Administrative')
print("number of outliers in Administrative: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'Administrative_Duration')
print("number of outliers in Administrative_Duration: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'Informational')
print("number of outliers in Informational: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'Informational_Duration')
print("number of outliers in Informational_Duration: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'ProductRelated')
print("number of outliers in ProductRelated: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'ProductRelated_Duration')
print("number of outliers in ProductRelated_Duration: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
```

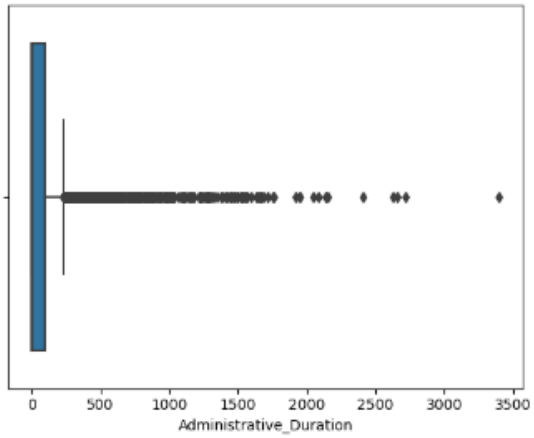
```
outliers = find_outliers(df, 'BounceRates')
print("number of outliers in BounceRates: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'ExitRates')
print("number of outliers in ExitRates: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'PageValues')
print("number of outliers in PageValues: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))

outliers = find_outliers(df, 'SpecialDay')
print("number of outliers in SpecialDay: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
```

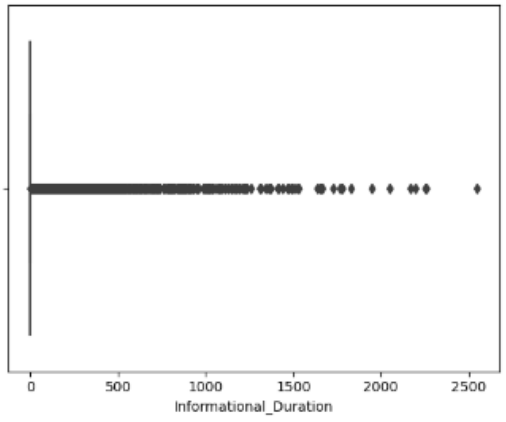
```
number of outliers in Administrative: 404
max outlier value: 27
min outlier value: 11
number of outliers in Administrative_Duration: 1172
max outlier value: 3398.75
min outlier value: 233.25
number of outliers in Informational: 2631
max outlier value: 24
min outlier value: 1
number of outliers in Informational_Duration: 2405
max outlier value: 2549.375
min outlier value: 1.0
number of outliers in ProductRelated: 987
max outlier value: 705
min outlier value: 85
number of outliers in ProductRelated_Duration: 961
max outlier value: 63973.52223
min outlier value: 3386.0
number of outliers in BounceRates: 1551
max outlier value: 0.2
min outlier value: 0.042083333
number of outliers in ExitRates: 1099
max outlier value: 0.2
min outlier value: 0.103703704
number of outliers in PageValues: 2730
max outlier value: 361.7637419
min outlier value: 0.038034542
number of outliers in SpecialDay: 1251
max outlier value: 1.0
min outlier value: 0.2
```

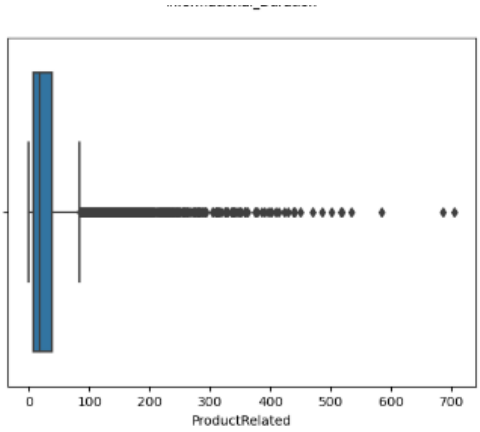
```
#boxplotting all variables showing outliers
boxplot=sns.boxplot(x='Administrative',data=df)
plt.show()
boxplot=sns.boxplot(x='Administrative_Duration',data=df)
plt.show()
boxplot=sns.boxplot(x='Informational',data=df)
plt.show()
boxplot=sns.boxplot(x='Informational_Duration',data=df)
plt.show()
boxplot=sns.boxplot(x='ProductRelated',data=df)
plt.show()
boxplot=sns.boxplot(x='ProductRelated_Duration',data=df)
plt.show()
boxplot=sns.boxplot(x='BounceRates',data=df)
plt.show()
boxplot=sns.boxplot(x='ExitRates',data=df)
plt.show()
boxplot=sns.boxplot(x='PageValues',data=df)
plt.show()
boxplot=sns.boxplot(x='SpecialDay',data=df)
plt.show()
```

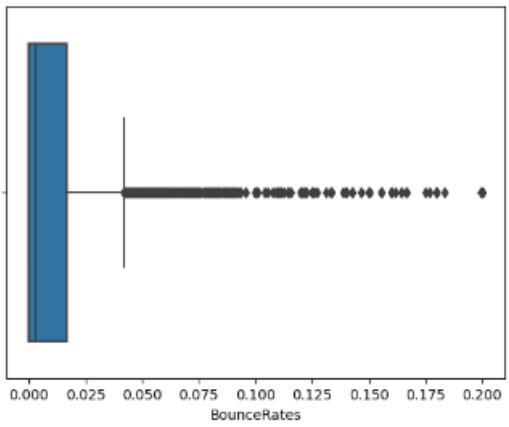


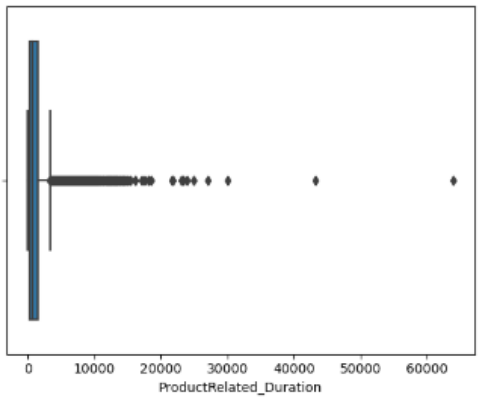


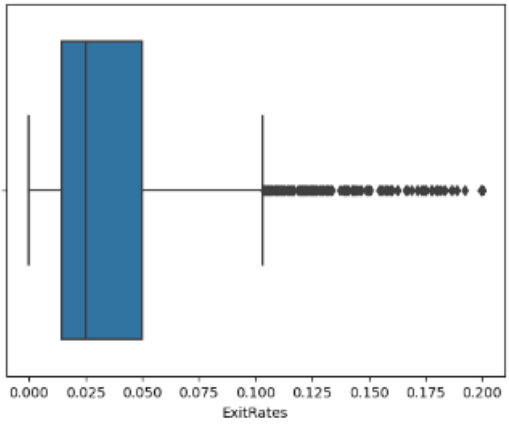


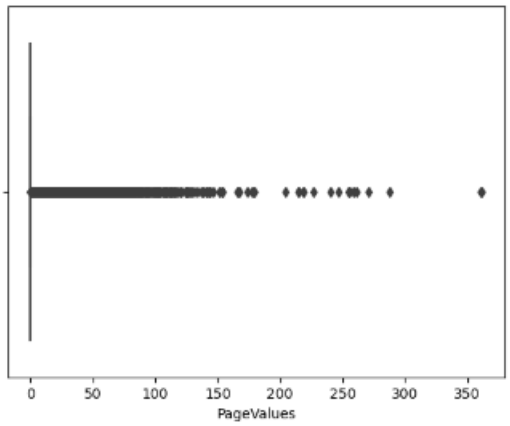


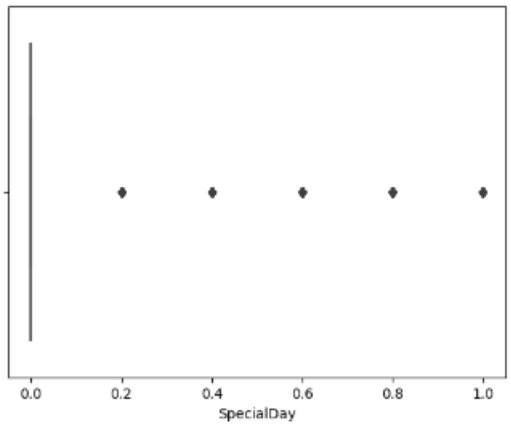

0
100
200
300
400
500
600
700
ProductRelated


0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
BounceRates


0
10000
20000
30000
40000
50000
60000
ProductRelated_Duration

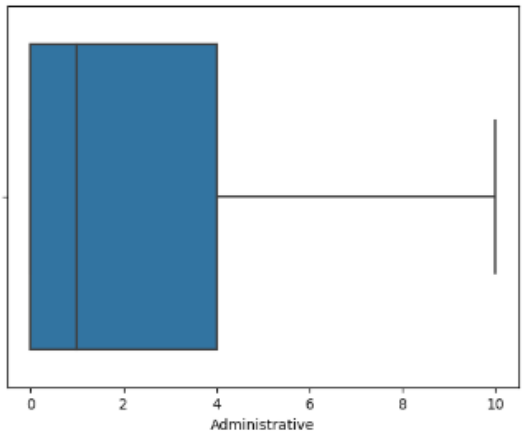

0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
ExitRates

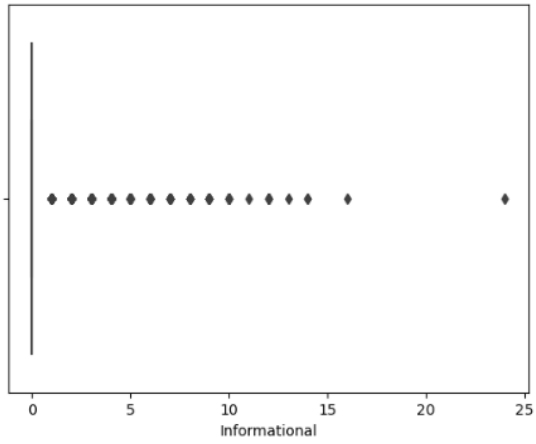

0
50
100
150
200
250
300
350
PageValues

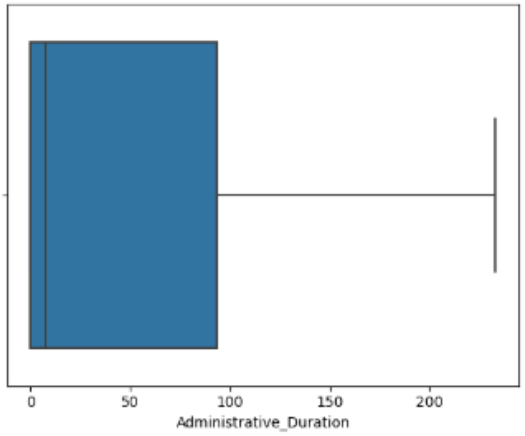

0.0
0.2
0.4
0.6
0.8
1.0
SpecialDay

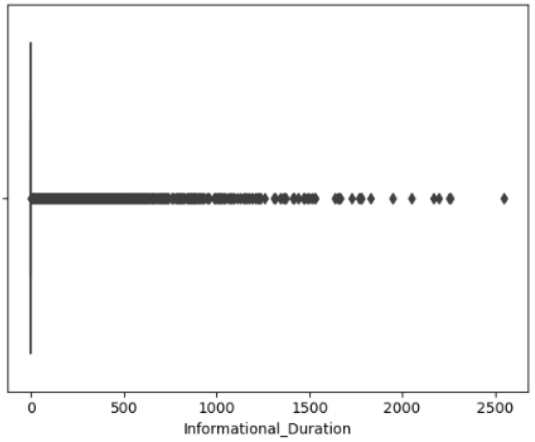
```
#treating outliers found.

def find_boundary(df, var):
    Q1 = df[var].quantile(0.25)
    Q3 = df[var].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1-(1.5*IQR)
    upper = Q3+(1.5*IQR)
    return lower , upper

lower_adm, upper_adm = find_boundary(df, 'Administrative' )
df.Administrative = np.where(df.Administrative > upper_adm, upper_adm,
                             np.where(df.Administrative < lower_adm, lower_adm, df.Administrative))

lower_adur, upper_adur = find_boundary(df, 'Administrative_Duration' )
df.Administrative_Duration = np.where(df.Administrative_Duration > upper_adur, upper_adur,
                             np.where(df.Administrative_Duration < lower_adur, lower_adur, df.Administrative_Duration))

lower_pr, upper_pr = find_boundary(df, 'ProductRelated' )
df.ProductRelated = np.where(df.ProductRelated > upper_pr, upper_pr,
                             np.where(df.ProductRelated < lower_pr, lower_pr, df.ProductRelated))

lower_prd, upper_prd = find_boundary(df, 'ProductRelated_Duration' )
df.ProductRelated_Duration = np.where(df.ProductRelated_Duration > upper_prd, upper_prd,
                             np.where(df.ProductRelated_Duration < lower_prd, lower_prd, df.ProductRelated_Duration))

lower_bou, upper_bou = find_boundary(df, 'BounceRates' )
df.BounceRates = np.where(df.BounceRates > upper_bou, upper_bou,
                             np.where(df.BounceRates < lower_bou, lower_bou, df.BounceRates))

lower_ex, upper_ex = find_boundary(df, 'ExitRates' )
df.ExitRates = np.where(df.ExitRates > upper_ex, upper_ex,
                             np.where(df.ExitRates < lower_ex, lower_ex, df.ExitRates))
```

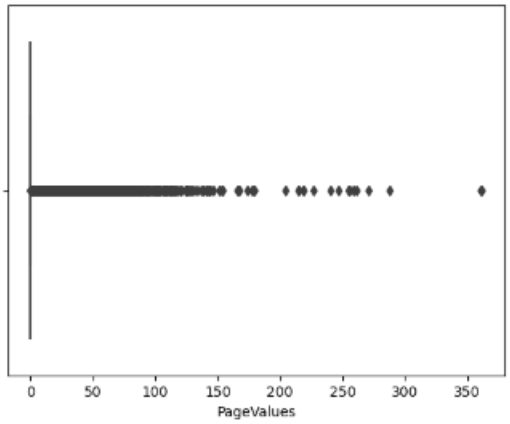
```
#re-boxplotting all variables showing outliers
boxplot=sns.boxplot(x='Administrative',data=df)
plt.show()
boxplot=sns.boxplot(x='Administrative_Duration',data=df)
plt.show()
boxplot=sns.boxplot(x='Informational',data=df)
plt.show()
boxplot=sns.boxplot(x='Informational_Duration',data=df)
plt.show()
boxplot=sns.boxplot(x='ProductRelated',data=df)
plt.show()
boxplot=sns.boxplot(x='ProductRelated_Duration',data=df)
plt.show()
boxplot=sns.boxplot(x='BounceRates',data=df)
plt.show()
boxplot=sns.boxplot(x='ExitRates',data=df)
plt.show()
boxplot=sns.boxplot(x='PageValues',data=df)
plt.show()
boxplot=sns.boxplot(x='SpecialDay',data=df)
plt.show()
```

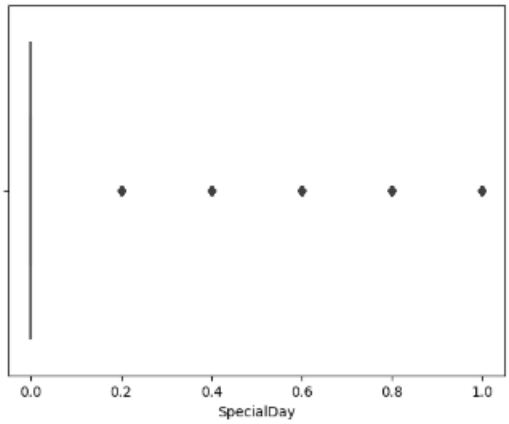

0
2
4
6
8
10
Administrative

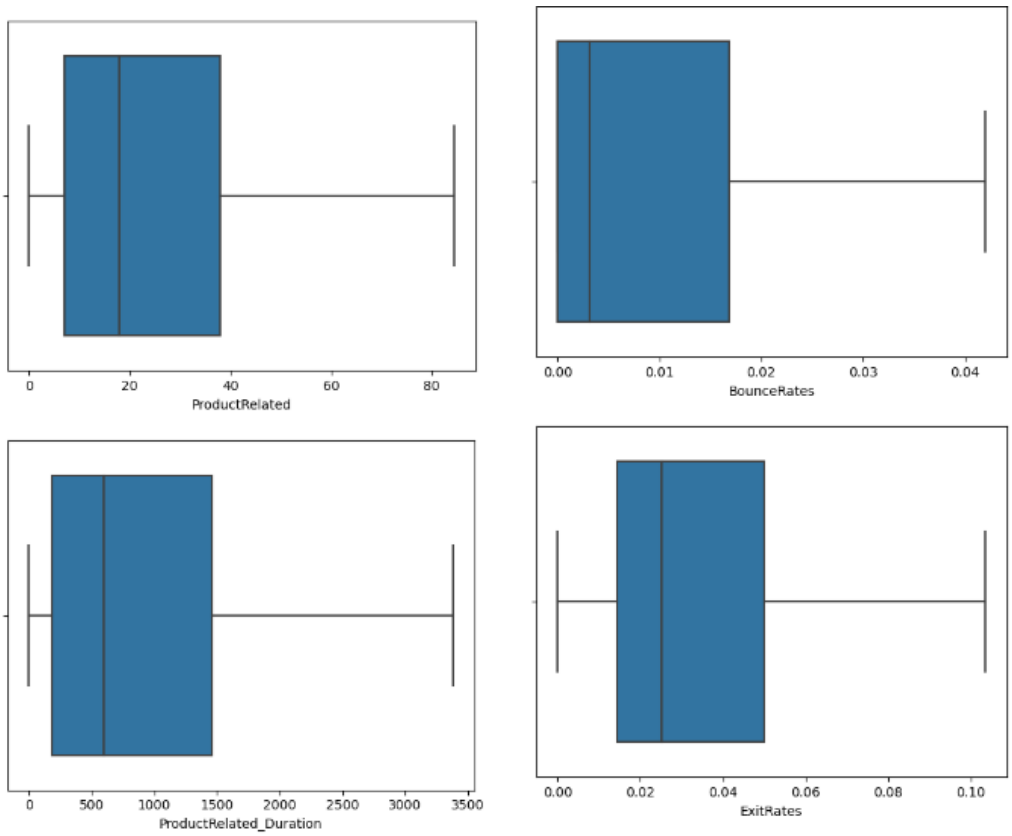

0
5
10
15
20
25
Informational


0
50
100
150
200
Administrative_Duration


0
500
1000
1500
2000
2500
Informational_Duration


0
50
100
150
200
250
300
350
PageValues


0.0
0.2
0.4
0.6
0.8
1.0
SpecialDay



The next part of the data preparation process was the data exploration phase. This phase began with a summary of statistics, identification of cardinality and reduction, and visualizing the univariate and bivariate statistics of the explanatory variables. Summary statistics were generated using the .describe() function to show multiple metrics. These metrics included the count, mean, standard deviation, minimum, maximum, and quantile measurements for each of the variables with an integer or float type.

```
#summary statistics
df.describe()
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates
count	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000
mean	2.221303	57.899459	0.508726	34.825454	26.835887	1001.332641	0.010568	0.035474
std	2.931051	80.206380	1.275617	141.424807	24.870099	1036.985399	0.014521	0.029252
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	8.000000	193.000000	0.000000	0.014231
50%	1.000000	9.000000	0.000000	0.000000	18.000000	608.942857	0.002899	0.025000
75%	4.000000	94.700000	0.000000	0.000000	38.000000	1477.154762	0.016667	0.048529
max	10.000000	236.750000	24.000000	2549.375000	83.000000	3403.386905	0.041667	0.099977

PageValues	SpecialDay	OperatingSystems	Browser	Region	TrafficType
12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000
5.949574	0.061942	2.124211	2.357804	3.153298	4.073904
18.653671	0.199666	0.906823	1.710114	2.402340	4.016654
0.000000	0.000000	1.000000	1.000000	1.000000	1.000000
0.000000	0.000000	2.000000	2.000000	1.000000	2.000000
0.000000	0.000000	2.000000	2.000000	3.000000	2.000000
0.000000	0.000000	3.000000	2.000000	4.000000	4.000000
361.763742	1.000000	8.000000	13.000000	9.000000	20.000000

Cardinality was visualized to help with the reduction of the dataset. This was completed via the usage of the .nunique() function. This function provided the total count of unique values within each variable. Using this information, the high cardinality and unneeded categorical variables were dropped. This included the following attributes: “Month”, “Region”, “OperatingSystems”, “Browser”, and “TrafficType”. All unneeded variables were dropped using the .drop() function.

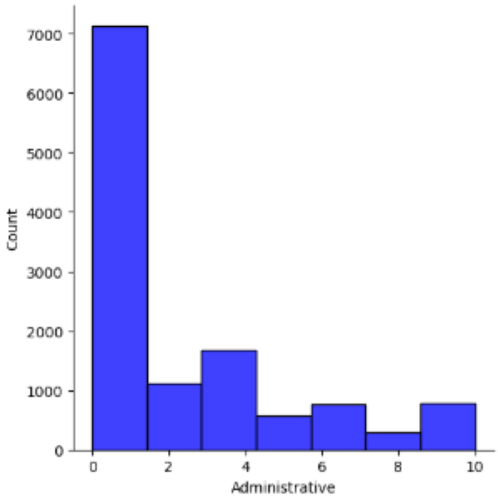
```
#locating cardinality of variables
print(f'Administrative: {df.Administrative.nunique()}')
print(f'Administrative_Duration: {df.Administrative_Duration.nunique()}')
print(f'Informational: {df.Informational.nunique()}')
print(f'Informational_Duration: {df.Informational_Duration.nunique()}')
print(f'ProductRelated: {df.ProductRelated.nunique()}')
print(f'ProductRelated_Duration: {df.ProductRelated_Duration.nunique()}')
print(f'BounceRates: {df.BounceRates.nunique()}')
print(f'ExitRates: {df.ExitRates.nunique()}')
print(f'PageValues: {df.PageValues.nunique()}')
print(f'SpecialDay: {df.SpecialDay.nunique()}')
print(f'Month: {df.Month.nunique()}')
print(f'OperatingSystems: {df.OperatingSystems.nunique()}')
print(f'Browser: {df.Browser.nunique()}')
print(f'Region: {df.Region.nunique()}')
print(f'TrafficType: {df.TrafficType.nunique()}')
print(f'VisitorType: {df.VisitorType.nunique()}')
print(f'Weekend: {df.Browser.nunique()}')
print(f'Revenue: {df.Region.nunique()}')
```

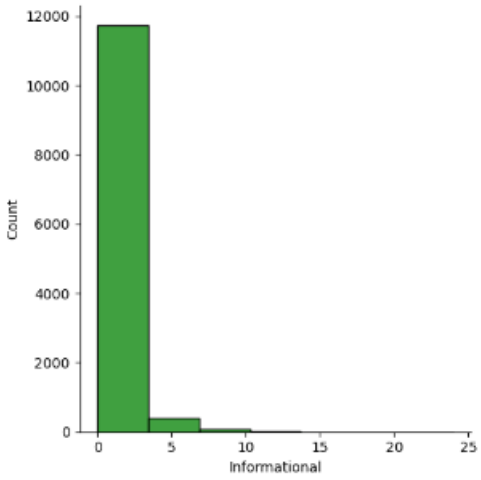
```
Administrative: 11
Administrative_Duration: 2258
Informational: 1
Informational_Duration: 1
ProductRelated: 84
ProductRelated_Duration: 8601
BounceRates: 1678
ExitRates: 4674
PageValues: 1
SpecialDay: 1
Month: 10
OperatingSystems: 8
Browser: 13
Region: 9
TrafficType: 20
VisitorType: 3
Weekend: 13
Revenue: 9
```

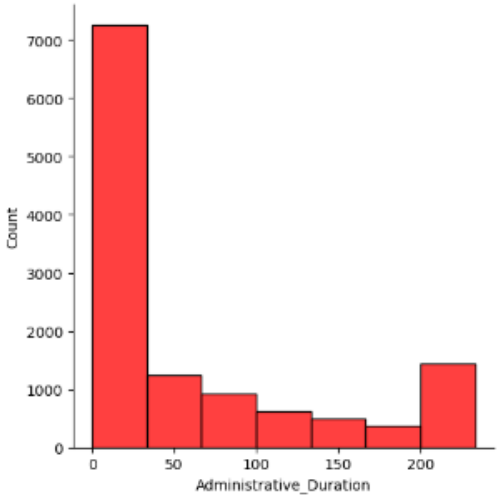
```
#dropping categorical unneeded variables
df.drop(['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType'], axis=1, inplace=True)
```

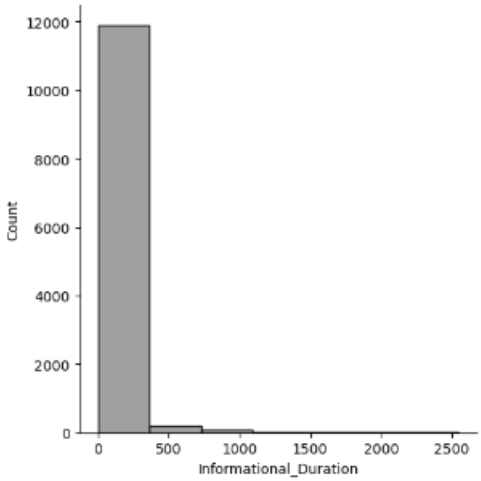
Univariate statistics of the quantitative variables were shown via .distplots from Seaborn. The following variables all had right skewed distributions: “Administrative”, “Administrative_Duration”, “ProductRelated”, “ProductRelated_Duration”, “BounceRates”, “ExitRates”, “Informational”, “Informational_Duration”, “PageValues”, and “SpecialDay”.

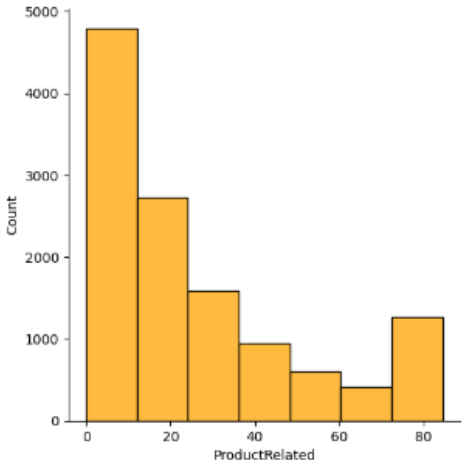
```
#univariate statistics, via distplots for quantitative explanatory variables

sns.displot(df['Administrative'], kde=False, color='blue', bins=7)
sns.displot(df['Administrative_Duration'], kde=False, color='red', bins=7)
sns.displot(df['Informational'], kde=False, color='green', bins=7)
sns.displot(df['Informational_Duration'], kde=False, color='gray', bins=7)
sns.displot(df['ProductRelated'], kde=False, color='orange', bins=7)
sns.displot(df['ProductRelated_Duration'], kde=False, color='black', bins=5)
sns.displot(df['BounceRates'], kde=False, color='purple', bins=5)
sns.displot(df['ExitRates'], kde=False, color='pink', bins=7)
sns.displot(df['PageValues'], kde=False, color='brown', bins=7)
sns.displot(df['SpecialDay'], kde=False, color='white', bins=7)
```

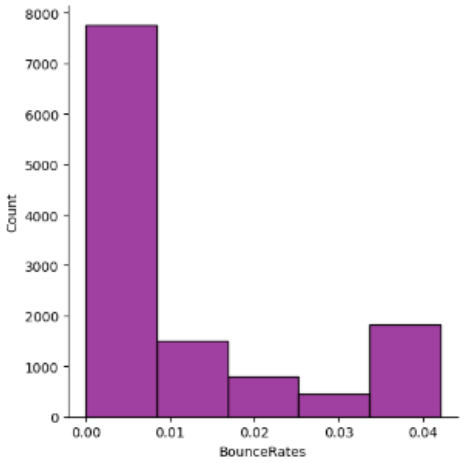


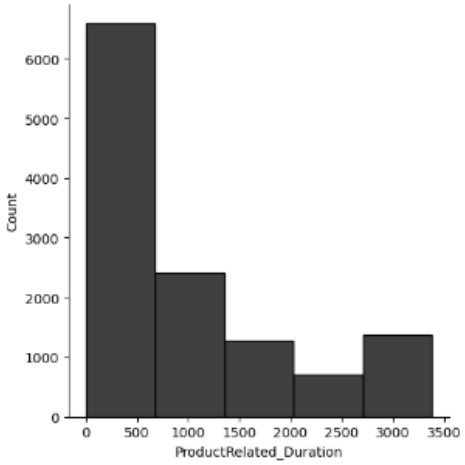


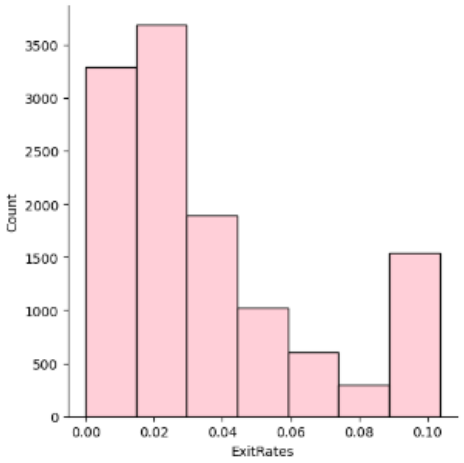


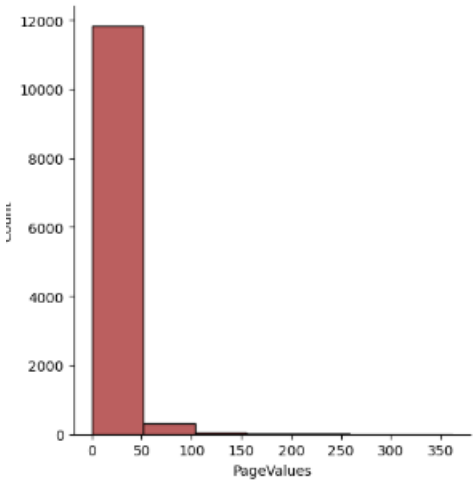


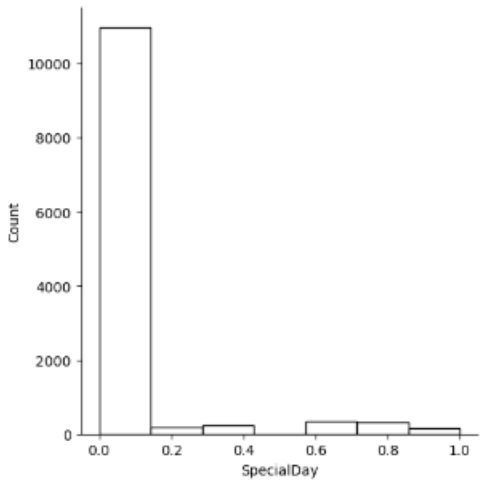

Count
ProductRelated


Count
BounceRates


Count
ProductRelated_Duration


Count
ExitRates

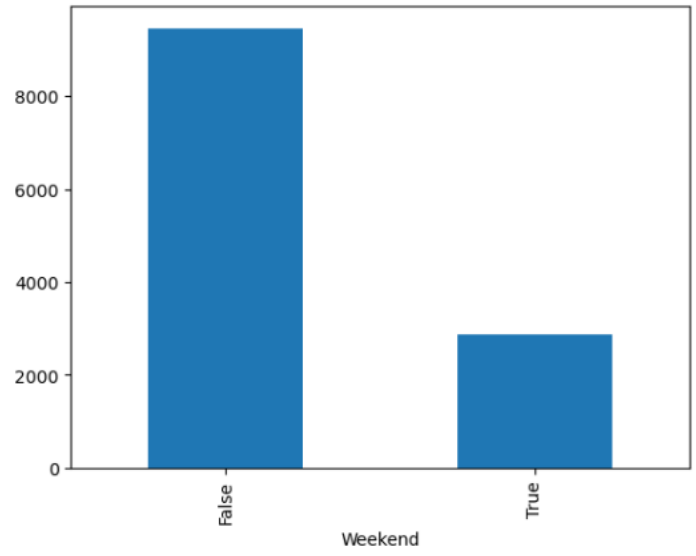

PageValues


Count
SpecialDay

The univariate statistics of the categorical explanatory were shown via bar plots. The variables' values were grouped and plotted to form the box plot. This was accomplished with the use of the .groupby() and .size() functions. The visualization of the data was provided via the Matplotlib library as an inline action.

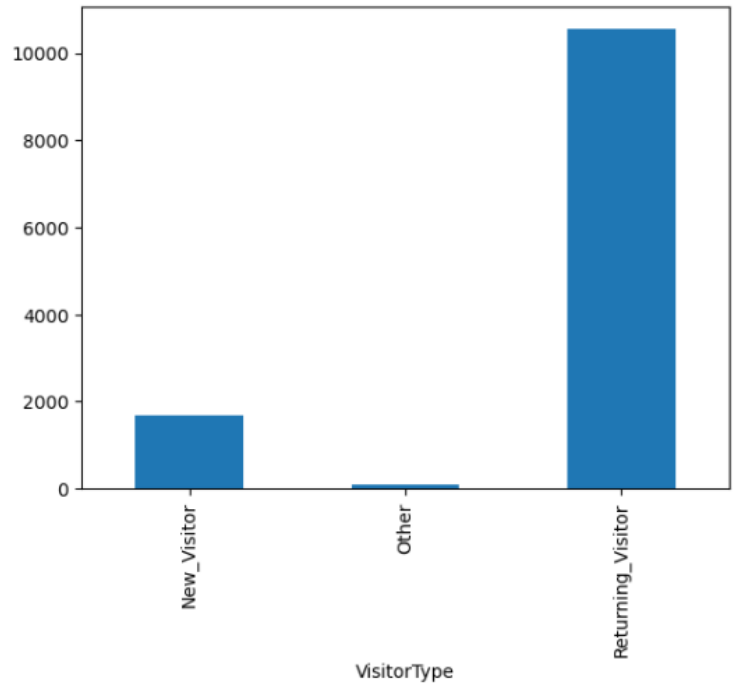
```
# univariate statistics, via barplots for categorical explanatory variables

groupedWeekEnd = df.groupby(by='Weekend').size()
groupedWeekEnd
%matplotlib inline
groupedWeekEnd.plot.bar()
```

```
<AxesSubplot: xlabel='Weekend'>
```



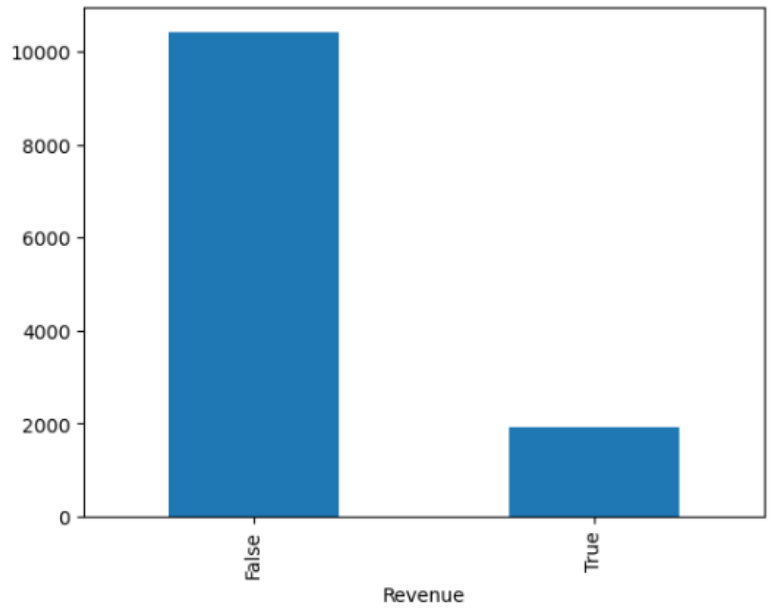
```
groupedVisitor = df.groupby(by='VisitorType').size()
groupedVisitor
%matplotlib inline
groupedVisitor.plot.bar()
```

```
<AxesSubplot: xlabel='VisitorType'>
```




```
groupedRevenue = df.groupby(by='Revenue').size()
groupedRevenue
%matplotlib inline
groupedRevenue.plot.bar()
```

```
<AxesSubplot: xlabel='Revenue'>
```



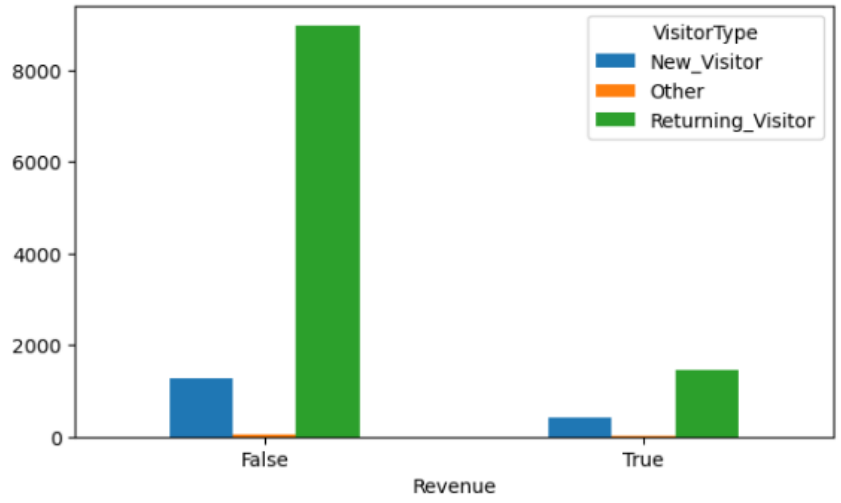
Bivariate statistics were represented in a couple of different ways. The target variable of "Revenue" was categorical. It was combined with the other categorical variables within the Pandas' .crosstab() function to form the first type of bivariate statistics. The cross-tabular table results were inputted into Matplotlib's bar plot.

```
#bivariate statistics visualizations
# 2 categorical variables using crosstab table and barplot

rv_crosstab=pd.crosstab(index=df['Revenue'], columns=df['VisitorType'])
print(rv_crosstab)
```

```
VisitorType  New_Visitor  Other  Returning_Visitor
Revenue
False               1271     65               8961
True                 422     16               1470
```

```
rv_crosstab.plot.bar(figsize=(7,4), rot=0)
```

```
<AxesSubplot: xlabel='Revenue'>
```

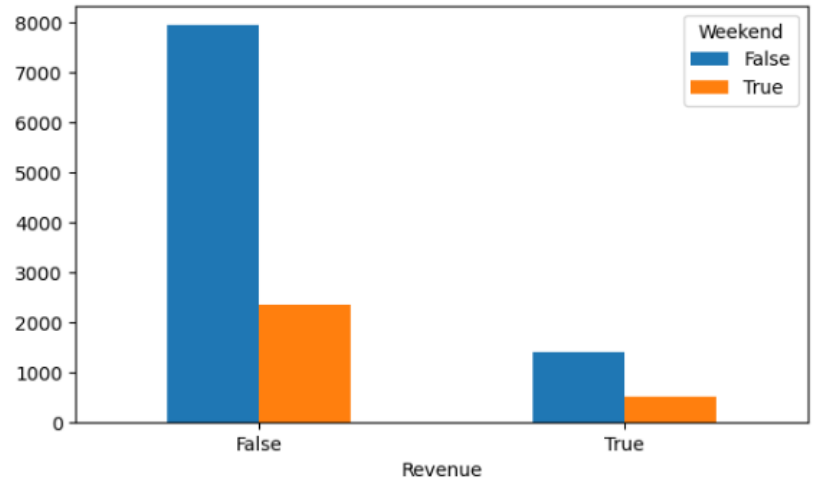


```
rw_crosstab=pd.crosstab(index=df['Revenue'], columns=df['Weekend'])
print(rw_crosstab)
```

```
Weekend  False  True
Revenue
False     7937  2360
True      1409   499
```

```
rw_crosstab.plot.bar(figsize=(7,4), rot=0)
```

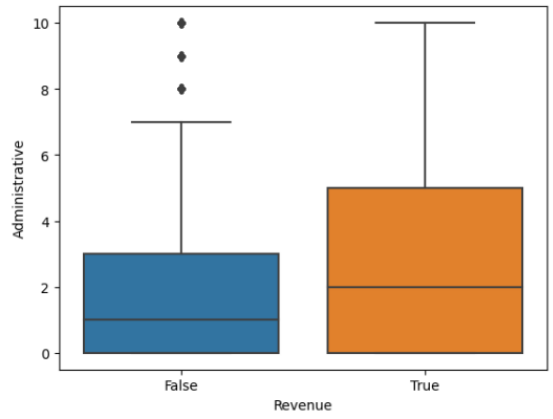
```
<AxesSubplot: xlabel='Revenue'>
```



The second type was combining the target variable of “Revenue” with the quantitative or continuous variables. This was shown with the use of the Seaborn .boxplot() function. The x-axis value was the target variable of “Revenue”. The y-axis values represented each of the explanatory variables.

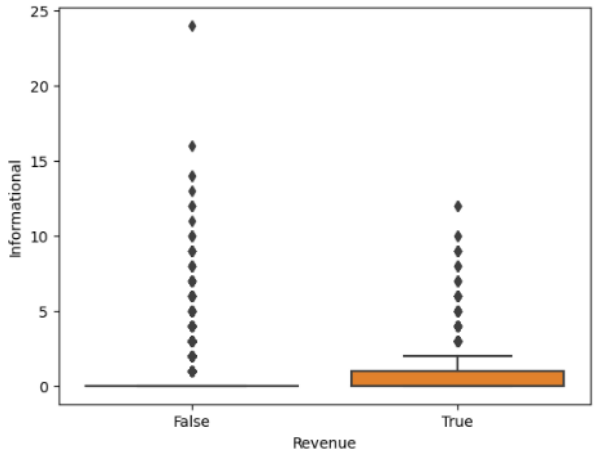
```
# bivariate statistics of categorical/continuous variable via boxplots
sns.boxplot(data=df, x="Revenue", y="Administrative")
```

```
<AxesSubplot: xlabel='Revenue', ylabel='Administrative'>
```



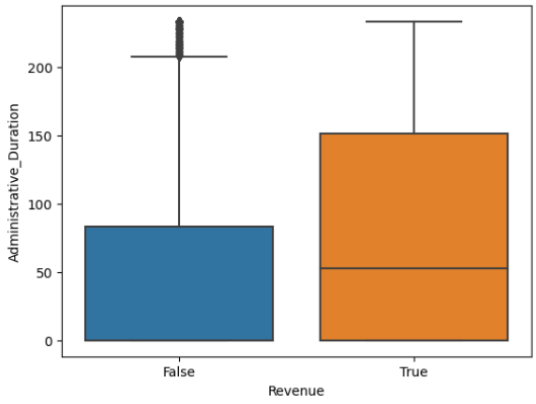

```
sns.boxplot(data=df, x="Revenue", y="Informational")
```

```
<AxesSubplot: xlabel='Revenue', ylabel='Informational'>
```



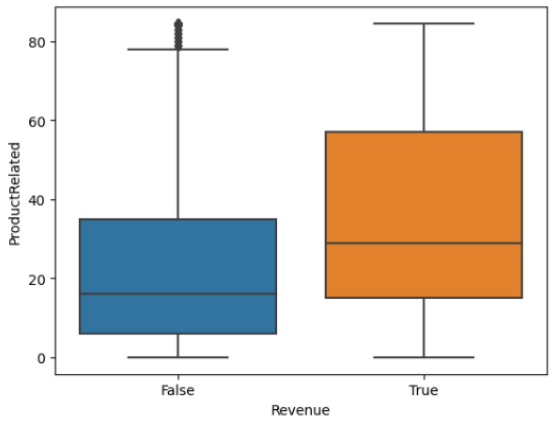

```
sns.boxplot(data=df, x="Revenue", y="Administrative_Duration")
```

```
<AxesSubplot: xlabel='Revenue', ylabel='Administrative_Duration'>
```



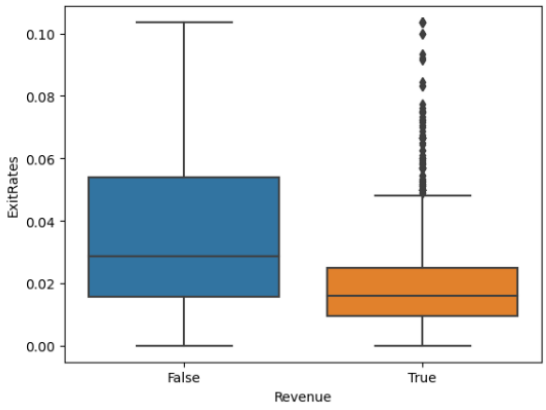

```
sns.boxplot(data=df, x="Revenue", y="ProductRelated")
```

```
<AxesSubplot: xlabel='Revenue', ylabel='ProductRelated'>
```



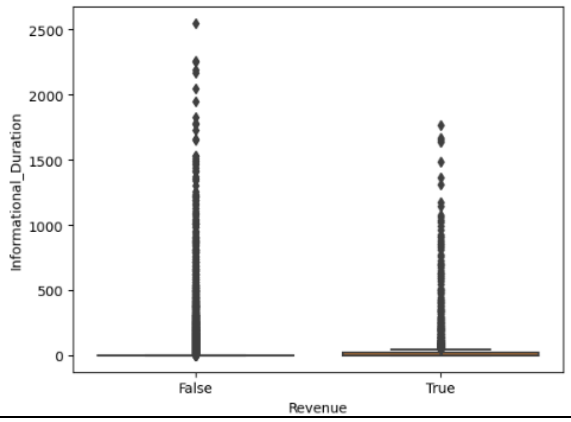

```
sns.boxplot(data=df, x="Revenue", y="ExitRates")
```

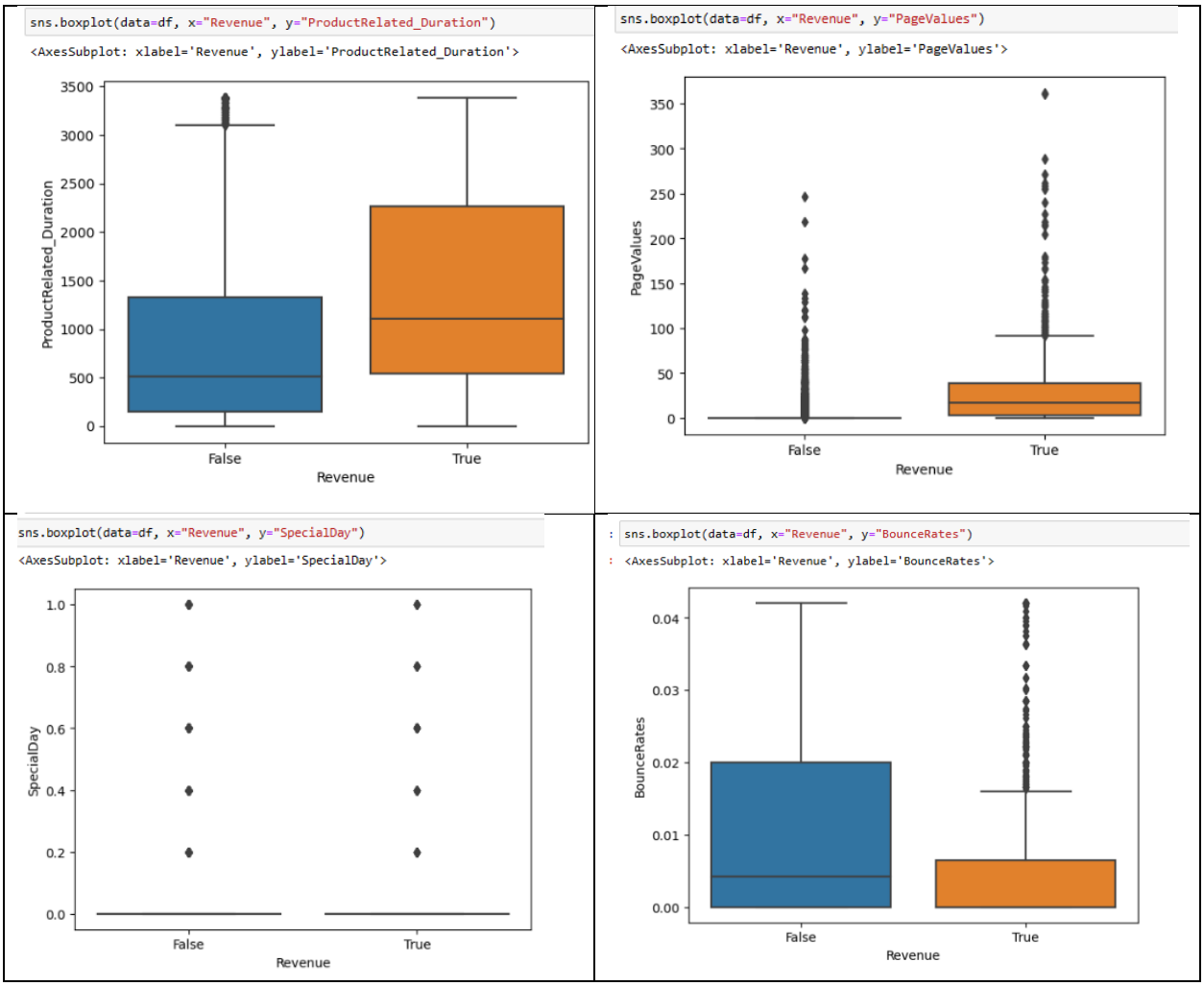
```
<AxesSubplot: xlabel='Revenue', ylabel='ExitRates'>
```




```
sns.boxplot(data=df, x="Revenue", y="Informational_Duration")
```

```
<AxesSubplot: xlabel='Revenue', ylabel='Informational_Duration'>
```





The final phase was data wrangling. In the data wrangling phase, the categorical variables were re-expressed into numerical variables. The categorical variables with only two unique values were converted with the use of ordinal encoding. It was used on the following variables, “Weekend” and “Revenue”. The ordinal encoding method utilized the cat.codes function to change each “False” value to “0” and each “True” to “1”.

```
#re-expression of categorical variables
#convert ordinal categorical to numerical

df['Weekend']=df['Weekend'].astype('category')
df['Weekend']=df['Weekend'].cat.codes
df['Revenue']=df['Revenue'].astype('category')
df['Revenue']=df['Revenue'].cat.codes
```

The categorical explanatory variables with more than two unique values were converted with the use of the Pandas' get_dummies method. The attributes were passed through the function to create dummy variables of the original attributes. This method concatenated the original variable name with each unique value within. This formed the new dummy variable with a "1" in the column of the dummy variable that was previously referenced by the original variable. For example, the variable "VisitorType" had three unique values of "New_Visitor", Returning_Visitor" and "Other". The new dummy variables created included "VisitorType New_Visitor", "VisitorType Other", and "VisitorType Returning_Visitor". After the completion of the get_dummies process, one of the dummy variables of each grouping was removed to reduce multicollinearity. In this dataset, the dummy variable of "VisitorType New_Visitor" was dropped.

```
#utilizing get_dummies to convert nominal categorical to numerical
df= pd.get_dummies(df, columns=['VisitorType'], prefix_sep=" " , drop_first=True)
```

The last step of the data wrangling procedure was to check for multicollinearity. The usage of the variance inflation factor (VIF) method was able to complete this task. A VIF value greater than ten had a high likelihood of multicollinearity. In this dataset, there were no variables with a VIF score of greater than ten. All variables were kept.

```
X=df[['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Weekend',
       'VisitorType Other', 'VisitorType Returning_Visitor']]

vif_data = pd.DataFrame()
vif_data["Explanatory Variables"] = X.columns

vif_data["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

vif_data["VIF"]=round(vif_data["VIF"],2)
vif_data=vif_data.sort_values(by="VIF", ascending=False)

print(vif_data)
```

```
            Explanatory Variables   VIF
4                   ProductRelated  8.76
5         ProductRelated_Duration  7.32
7                        ExitRates  5.76
12  VisitorType Returning_Visitor  5.46
0                   Administrative  5.23
1         Administrative_Duration  4.61
6                     BounceRates  3.34
2                   Informational  2.11
3          Informational_Duration  1.73
10                        Weekend  1.26
9                      SpecialDay  1.14
8                      PageValues  1.12
11              VisitorType Other  1.03
```

Part IV. Data Analysis

D. Discussion of Data Analysis Process

The logistic regression method was deemed the appropriate technique to analyze the presented research question, “What attributes within the captured Google Analytics metrics can contribute to the possibility of a customer purchasing an item?”. Logistic Regression was described as a statistical algorithm, which was used in binary classification analysis (IBM, n.d.). The goal of logistic regression was to predict the binary target variable outcome based on the explanatory variables inputted. In this online purchasing dataset, “Revenue” was selected as the target variable as a “True or False” binary value. The explanatory variables provided insight into the probability of the target variable occurring. One of the main advantages of logistic regression was its interpretable results. The logistic regression model provides coefficients that could be interpreted. The

coefficients represented the strength of the relationships between the target and explanatory variables. One disadvantage of logistic regression was feature selection. The regression model required careful selection of variables to avoid multicollinearity as it would affect the stability and skew interpretation of the model.

The initial logistic regression model was computed with the use of Statmodels' .logit() function on the target variable "Revenue" which was noted as "y" and the explanatory variables notated as "X". The explanatory variables included "Administrative", "Administrative_Duration", "Informational", "Informational_Duration", "ProductRelated", "ProductRelated_Duration", "BounceRates", "ExitRates", "PageValues", "SpecialDay", "Weekend", "VisitorType Other", "VisitorType Returning_Visitor", and "const". These results were fitted and printed to show the resulting summary.

```
#logistic regression model
df['const']=1
y= df['Revenue']
X= df[['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Weekend',
       'VisitorType Other', 'VisitorType Returning_Visitor','const']]
log_model = sm.Logit(y,X)
results = log_model.fit()
print(results.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.302019
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                12205
Model:                          Logit   Df Residuals:                    12191
Method:                           MLE   Df Model:                           13
Date:                Sat, 15 Jul 2023   Pseudo R-squ.:                 0.3034
Time:                        11:45:32   Log-Likelihood:                -3686.1
converged:                       True   LL-Null:                       -5291.3
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Administrative                   0.0052      0.017      0.310      0.756      -0.027       0.038
Administrative_Duration         -0.0003      0.001     -0.463      0.644      -0.001       0.001
Informational                    0.0204      0.026      0.787      0.431      -0.030       0.071
Informational_Duration           0.0002      0.000      0.747      0.455      -0.000       0.001
ProductRelated                   0.0047      0.002      2.095      0.036       0.000       0.009
ProductRelated_Duration          0.0003   5.13e-05      5.188      0.000       0.000       0.000
BounceRates                    -11.4239      4.365     -2.617      0.009     -19.979      -2.869
ExitRates                      -12.4520      2.410     -5.167      0.000     -17.175      -7.729
PageValues                       0.0794      0.002     33.910      0.000       0.075       0.084
SpecialDay                      -0.8830      0.216     -4.086      0.000      -1.307      -0.459
Weekend                          0.1431      0.070      2.043      0.041       0.006       0.280
VisitorType Other               -0.7309      0.521     -1.404      0.160      -1.751       0.289
VisitorType Returning_Visitor   -0.4543      0.088     -5.155      0.000      -0.627      -0.282
const                           -2.0319      0.098    -20.722      0.000      -2.224      -1.840
================================================================================================
```

After this was computed, the initial model noted the following evaluation metrics such as Log-Likelihood, Pseudo R squared, and LLR-p-value. Each of these metrics provided the regression model goodness of fit. The higher the value for Log-likelihood and Pseudo R squared the better the goodness of fit. For the initial model, these values were “-3686.1” for Log-likelihood and about .3034 or 30.34% for Pseudo R squared. The LLR p-value was the p-value for the complete model. Statistical significance was noted when the value was less than 0.05. In this model, the value was “0.0” deeming the model statistically significant.

The next phase of the analysis was assessing the performance and accuracy of the logistic regression model. Using the same variables, the data was manipulated using SciKit Learn’s train_test_split method. The dataset was split into two datasets, one being training data and the other testing data. This was split in a 70:30 ratio. The testing data consisted of 30% of the original dataset, with the remaining 70% going into the training dataset. The explanatory variables within the training data were scaled with the use of SciKit Learn’s RobustScaler. This

was then fit and transformed. The target variable within the testing dataset was also scaled and transformed as well. Logistic Regression was computed on the fitted target and explanatory variables from the training datasets. The resulting log regression model was predicted using the explanatory variables within the testing data.

```
y= df['Revenue']
X= df[['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Weekend',
       'VisitorType Other', 'VisitorType Returning_Visitor','const']]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_sta
```

```
scaler = RobustScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

```
X_test= scaler.transform(X_test)
```

```
log_r= LogisticRegression(random_state=0).fit(X_train, y_train)
```

```
y_pred= log_r.predict(X_test)
```

All of these calculations allowed for the accuracy score and confusion matrix array to be populated. The logistic regression score was 88.13% on the training dataset. The confusion matrix array resulted in True Positives, True Negatives, False Positives, and False Negatives. Those results were as follows:

True positive predictions: 3035

True negative predictions: 208

False positive predictions: 352

False negative predictions: 67

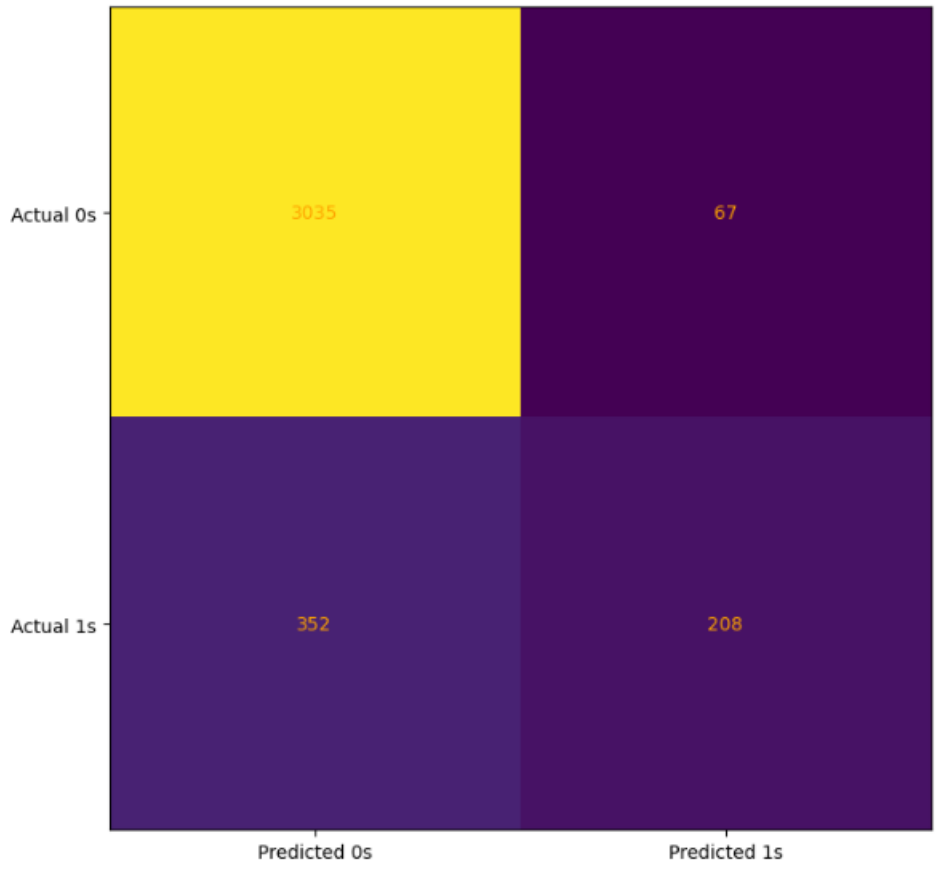
```
log_r.score(X_train, y_train)
```

```
0.8813063326700222
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[3035,   67],
       [ 352,  208]], dtype=int64)
```

```
matrix_i = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(matrix_i)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, matrix_i[i, j], ha='center', va='center', color='orange')
plt.show()
```



The last calculation was the accuracy evaluation metric. This was provided by the use of the classification report function. The initial model had a prediction accuracy score of 89%.

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      3102
           1       0.76      0.37      0.50       560

    accuracy                           0.89      3662
   macro avg       0.83      0.67      0.72      3662
weighted avg       0.87      0.89      0.87      3662
```

After the initial model was completed, an attempt to reduce the logistic regression model was made. This was completed with the usage of a wrapper method on the initial model. The wrapper method used was the Backward Stepwise Elimination method. This selection method utilized the initial regression model to review the p-values of the explanatory variables. The p-value was found in the P> |z| column. The elimination method was justified as it kept only the explanatory variables that had a p-value that was deemed statistically significant, which was noted as a lesser than 0.05 value. The variables which met this criterion were “ProductRelated”, “ProductRelated_Duration”, “BounceRates”, “ExitRates”, “PageValues”, “SpecialDay”, “Weekend”, “VisitorType Returning_Visitor”, and “const” The same logit function was reassessed using only these attributes along with the target variable. The reduced model calculated the following results, the Log-likelihood value was “-3688.7” and about “.3029” or 30.29% for Pseudo R squared. The LLR p-value stayed static at “0.0”.

```
#reduced model via backward stepwise elmination with p-values less than 0.05

df['const']=1
y= df['Revenue']
X=df[['ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Weekend',
       'VisitorType Returning_Visitor','const']]
log_model_red = sm.Logit(y,X)
redu_results = log_model_red.fit()
print(redu_results.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.302230
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                12205
Model:                          Logit   Df Residuals:                    12196
Method:                           MLE   Df Model:                            8
Date:                Sat, 15 Jul 2023   Pseudo R-squ.:                  0.3029
Time:                        11:46:51   Log-Likelihood:                -3688.7
converged:                       True   LL-Null:                       -5291.3
Covariance Type:            nonrobust   LLR p-value:                     0.000
=================================================================================================
                                    coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
ProductRelated                    0.0051      0.002      2.329      0.020       0.001       0.009
ProductRelated_Duration           0.0003   5.05e-05      5.392      0.000       0.000       0.000
BounceRates                     -11.1620      4.352     -2.565      0.010     -19.692      -2.632
ExitRates                       -12.5706      2.379     -5.285      0.000     -17.232      -7.909
PageValues                        0.0795      0.002     34.035      0.000       0.075       0.084
SpecialDay                       -0.8830      0.215     -4.098      0.000      -1.305      -0.461
Weekend                           0.1486      0.070      2.124      0.034       0.012       0.286
VisitorType Returning_Visitor    -0.4317      0.087     -4.949      0.000      -0.603      -0.261
const                            -2.0585      0.094    -21.923      0.000      -2.243      -1.874
=================================================================================================
```

The reduced model was processed through the same calculations for the train test split as well as the accuracy score and confusion matrix. This provided a comparison between the two models. The reduction model had a logistic regression score of 88.47% on the training dataset.

```
y= df['Revenue']
X=df[['ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Weekend',
       'VisitorType Returning_Visitor','const']]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

```
scaler = RobustScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

```
X_test= scaler.transform(X_test)
```

```
logr= LogisticRegression(random_state=0).fit(X_train, y_train)
```

```
logr.score(X_test, y_test)
```

```
0.8847624249044238
```

```
y_predr= logr.predict(X_test)
```

The confusion matrix for the reduced model computed the following results:

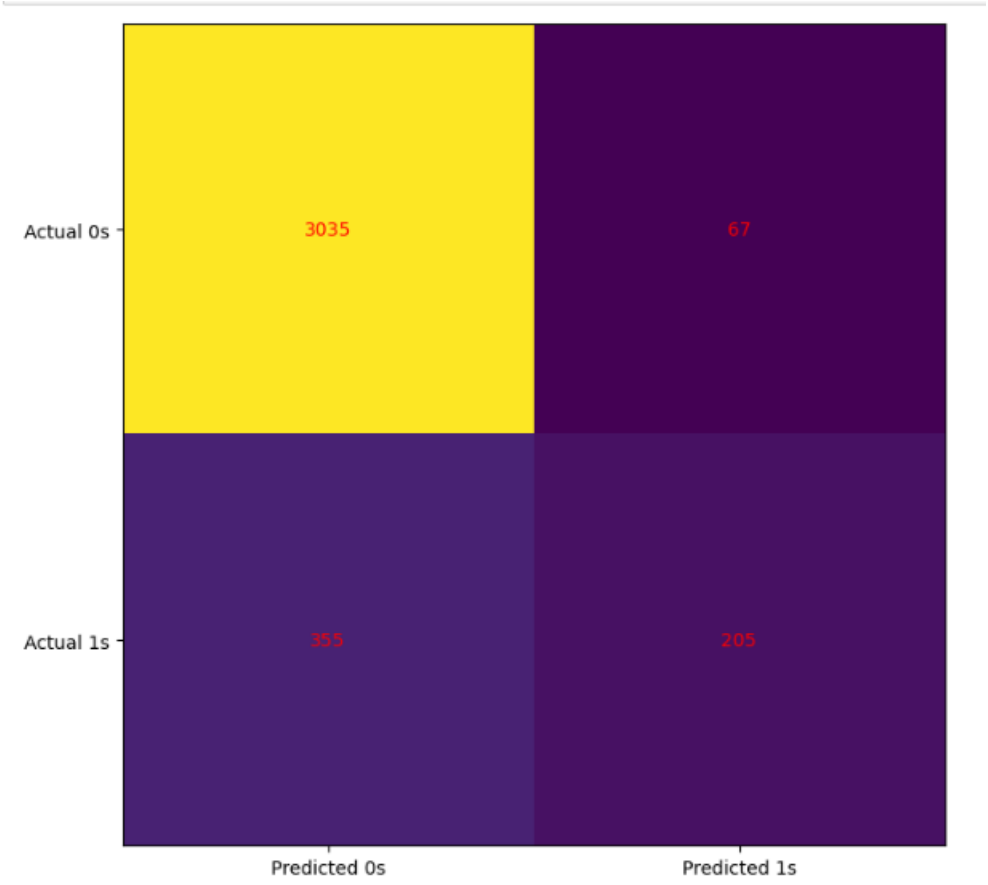
True positive predictions: 3035

True negative predictions: 205

False positive predictions: 355

False negative predictions: 67

```
matrix = confusion_matrix(y_test, y_predr)
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(matrix)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, matrix[i, j], ha='center', va='center', color='red')
plt.show()
```



Lastly, the model prediction accuracy score was 88% for the reduced model. This was shown in the below classification report summary.

```
print(classification_report(y_test, y_predr))
```

	precision	recall	f1-score	support
0	0.90	0.98	0.93	3102
1	0.75	0.37	0.49	560
accuracy			0.88	3662
macro avg	0.82	0.67	0.71	3662
weighted avg	0.87	0.88	0.87	3662

Part IV. Data Summary and Implications

E. Summarization of Findings

The research question for this capstone project was "What attributes within the captured Google Analytics metrics can contribute to the possibility of a customer purchasing an item?". The analysis began with the null and alternative hypotheses.

Null hypothesis: The logistic regression model shows that the predictor variable of "Revenue" and the explanatory variables do not have a statistically significant relationship.

Alternate Hypothesis: The logistic regression model shows a statistically significant relationship between the predictor variable "Revenue" and explanatory variables.

The final models of this logistic regression modeling resulted in similar outcomes. The accuracy scores for their predictions were both above the noted 80% threshold. The reduced model noted statistical significance in the following explanatory variables: "ProductRelated", "ProductRelated_Duration", "BounceRates", "ExitRates", "PageValues", "SpecialDay", "Weekend", "VisitorType Returning_Visitor". The variables' strength in prediction was identified based on their z-statistics. The z-statistics of the reduced model provided a better statistical fit due to the strength of the predictors. The z column in the regression summary noted

the z-statistics. When the z-statistic value was further from zero, the stronger its role was as a predictor variable. This was shown in “ProductRelated_Duration”, “ExitRates”, “PageValues”, and “VisitorType Returning_Visitor”. The strongest was noted as “PageValues” with a z-statistic value of “34.035”. As a result, the null hypothesis could be rejected in favor of the alternative hypothesis. This would note the logistic regression model could be effective in predicting binary outcomes and observed relationships between the target and explanatory variables.

There were some limitations to this analysis that should be noted. One of the biggest limitations was the lack of a specific timeframe. E-commerce has grown by leaps and bounds since the COVID-19 pandemic. According to census data, e-commerce sales surged during the pandemic (Brewster, 2022). With the lack of a particular year, there was no way to determine if current yearly data would have the same contributing attributes and resulting outcomes.

As for the course of action based on these results. The e-commerce company should investigate which website pages contributed to its revenue based on the page values associated. It could be identified as product-related based on the strength of the duration variable, but further investigation should be initiated (Google, n.d.). This could lead to additional strategic ideas for the sales and marketing teams. Also, with the strength of returning users, there could be additional client retention efforts put forth. This could be specific discounts or other marketing gimmicks.

There are some worthy analyses to be performed with this dataset. It could be beneficial to join it with an additional dataset to provide supplementary attributes. Another opportunity would be to utilize another categorical variable within predictive modeling to see if the same explanatory variables were statistically significant to it as well.

Lastly, another research analysis could be performed utilizing the regional or monthly data in regard to revenue. Different regions have different advantages in terms of speed and cost. Analysis of this attribute could provide additional insight to stakeholders. The same could be said for various times of the year. For example, the holiday season from November – January tends to have an increase in sales and revenue across most markets. It would be an ideal comparison of in person retail and online purchasing.

F. References

Bobbitt, Z. (2020, October 13). *The 6 Assumptions of Logistic Regression*. Retrieved from Statology: https://www.statology.org/assumptions-of-logistic-regression/

Google. (n.d.). *Analytics dimensions and metrics*. Retrieved from Google: https://support.google.com/analytics/answer/9143382?hl=en&sjid=5717273188166762214-NA

IBM. (n.d.). *What is logistic regression?* Retrieved from IBM: https://www.ibm.com/topics/logistic-regression

Joy, A. (n.d.). *5 Main Disadvantages of Python Programming Language*. Retrieved from Pythonista Planet: https://pythonistaplanet.com/disadvantages-of-python/#:~:text=The%20main%20disadvantages%20of%20Python%20are%20its%20slowness%20during%20execution,in%20the%20enterprise%20development%20sector.

Kumar, B. (2023, March 6). *Google Analytics for Ecommerce in 2023 (Complete Guide)*. Retrieved from Shopify: https://www.shopify.com/blog/14681601-google-analytics-for-ecommerce-a-beginners-guide

NumFOCUS, Inc. (2023, June 28). *pandas documentation*. Retrieved from pandas: https://pandas.pydata.org/pandas-docs/stable/reference/options.html

Sakar, C., & Kastro, Y. (2018). *Online Shoppers Purchasing Intention Dataset*. Retrieved from UCI Machine Learning Repository: https://doi.org/10.24432/C5F88Q

Sjursen, S. (2020, March 1). *The Pros and Cons of Using Jupyter Notebooks as Your Editor for Data Science Work*. Retrieved from Better Programming: https://betterprogramming.pub/pros-and-cons-for-jupyter-notebooks-as-your-editor-for-data-science-work-tip-pycharm-is-probably-40e88f7827cb

Statistic Solutions. (n.d.). *Assumptions of Logistic Regression*. Retrieved from https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/