APPLIED DATA ANALYTICS

Module Code: B8IT160

Name: Sean Carroll

Student Number: 20024157

Table of Contents

Introduction 3

Exploratory Analysis 4

Data Preprocessing 9

Predictive Analysis 16

Conclusion 20

References 21

Introduction

The Online Shoppers Purchasing Intention Dataset (Sakar & Kastro, 2018), sourced from UCI Machine Learning Repository consists of 17 features and 12,330 instances. The dataset describes online shoppers' sessions on an e-commerce platform over the course of a 1-year period. As per UCI Machine Learning Repository (n.d.), of the 12,330 sessions in the dataset, 10,422 did not result in shopping, and 1,908 did result in shopping. The dataset contains 6 continuous features and 11 categorical features. The 'Revenue' feature acts as the target variable. This report describes the exploratory analysis of the Online Shoppers Purchasing Intention Dataset, how the data within the dataset was preprocessed, and how this preprocessed data was used in predictive analysis.

Exploratory Analysis

Describe and discuss the dataset and features using appropriate graphs and tables

The df.describe() function displays the following table to describe the count, mean, standard deviation, minimum value, maximum value, 1st quartile, 3rd quartile, and median value of each column. They can be seen in the tables below.

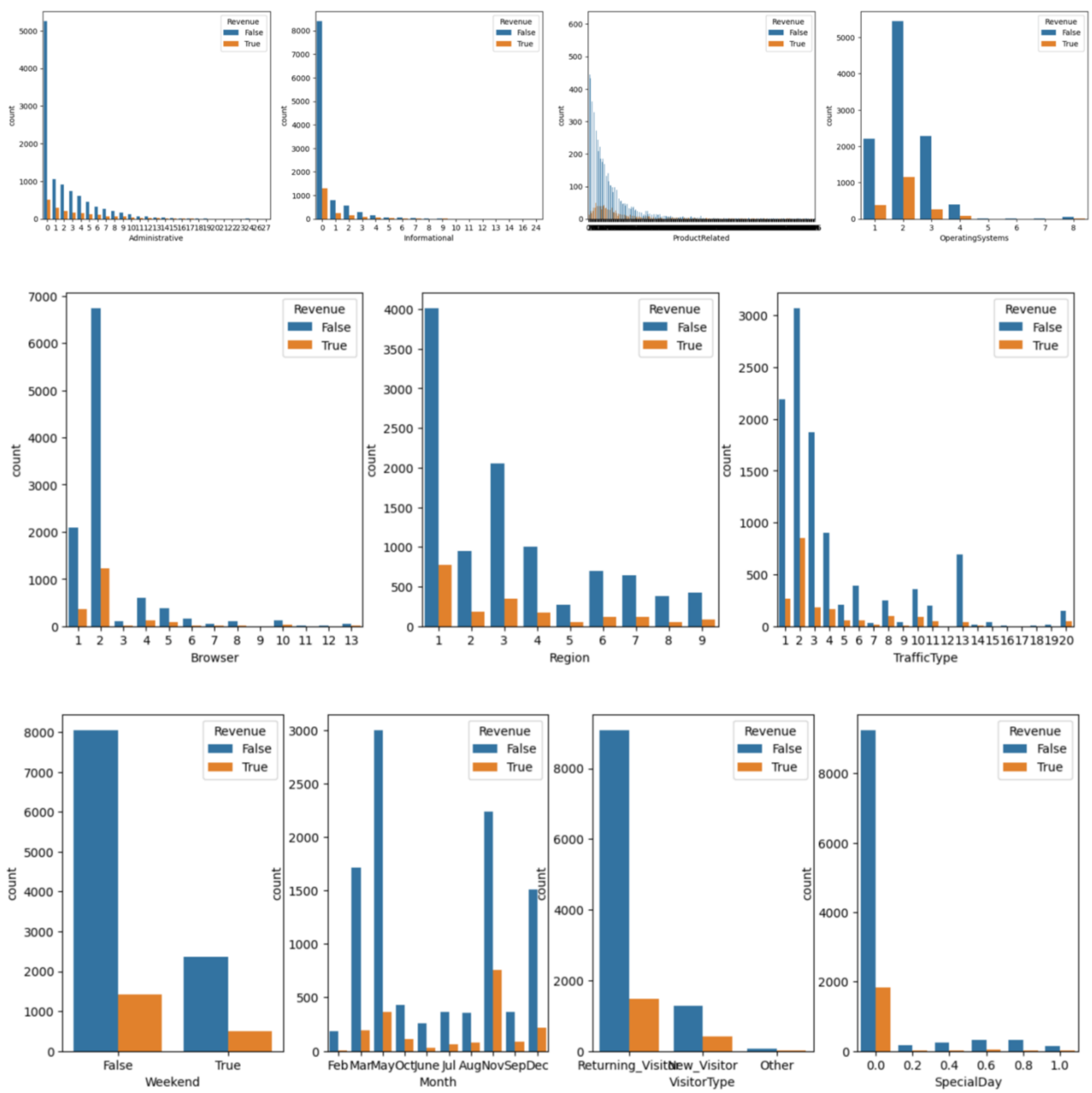
	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000
mean	2.315166	80.818611	0.503569	34.472398	31.731468	1194.746220	0.022191	0.043073
std	3.321784	176.779107	1.270156	140.749294	44.475503	1913.669288	0.048488	0.048597
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	7.000000	184.137500	0.000000	0.014286
50%	1.000000	7.500000	0.000000	0.000000	18.000000	598.936905	0.003112	0.025156
75%	4.000000	93.256250	0.000000	0.000000	38.000000	1464.157214	0.016813	0.050000
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	0.200000

PageValues	SpecialDay	OperatingSystems	Browser	Region	TrafficType
12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000
5.889258	0.061427	2.124006	2.357097	3.147364	4.069586
18.568437	0.198917	0.911325	1.717277	2.401591	4.025169
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

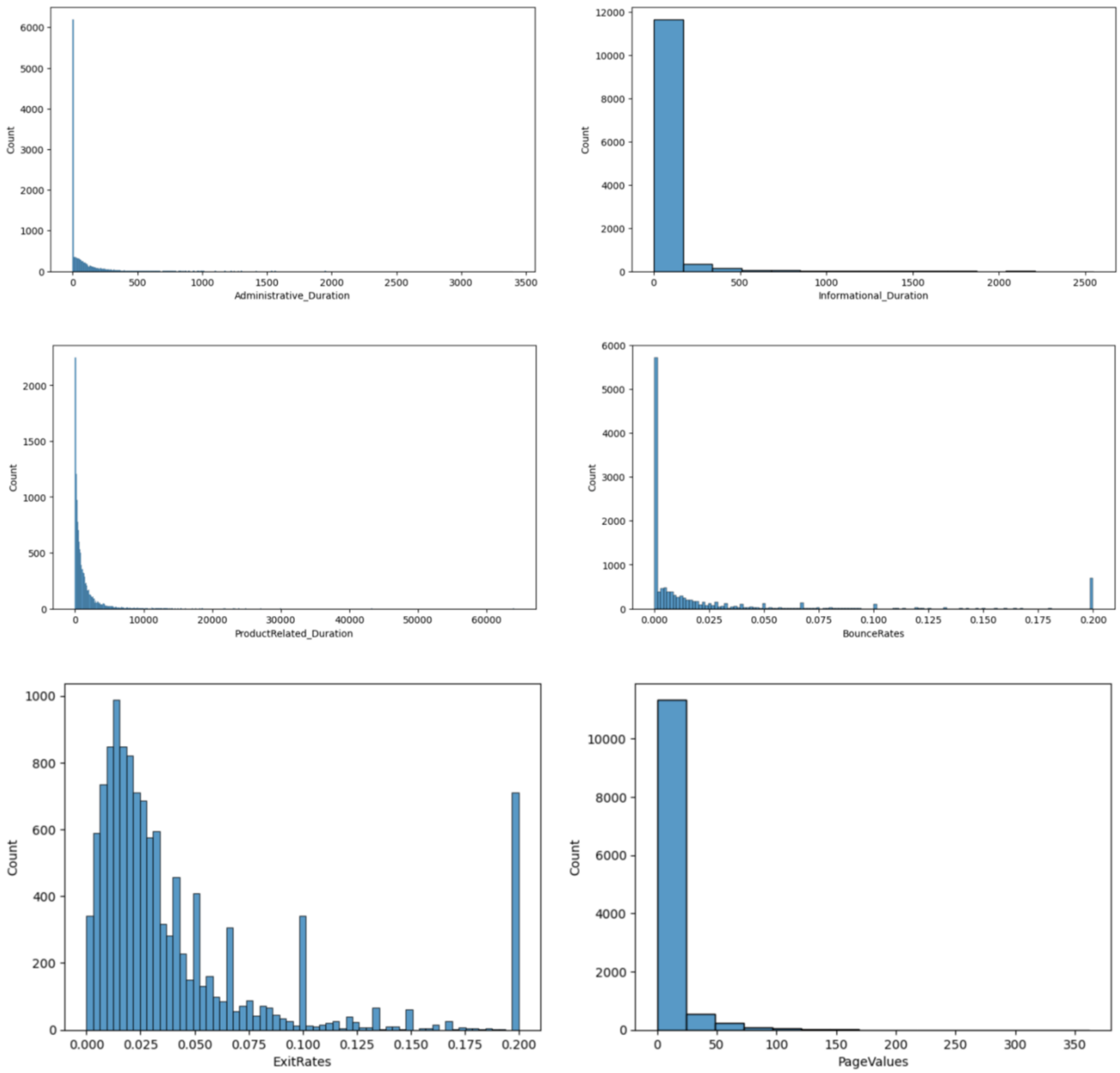
The df.info() function displays each of the features within the DataFrame, the number of null values in each feature, and the type of data in each feature. As per the below data, there are 17 features, none of which contain null values, and their corresponding data types.

```
<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
```

The bar charts below describe the discrete and categorical data in the DataFrame. While the ‘SpecialDay’ feature appeared to be continuous data in the above table described by the df.info() function it is actually categorical data, with each special day being categorically assigned a float value. As seen in the graphs the ‘OperatingSystems’ feature is the only feature with a somewhat normal distribution, and the ‘Month’ feature has a somewhat bimodal distribution. The other 9 features appear to have a positively skewed distribution.



The histograms below describe the continuous data in the DataFrame. Like the majority of the categorical features, 5 of the continuous features have a positively skewed distribution. The ‘ExitRates’ feature has a bimodal distribution.



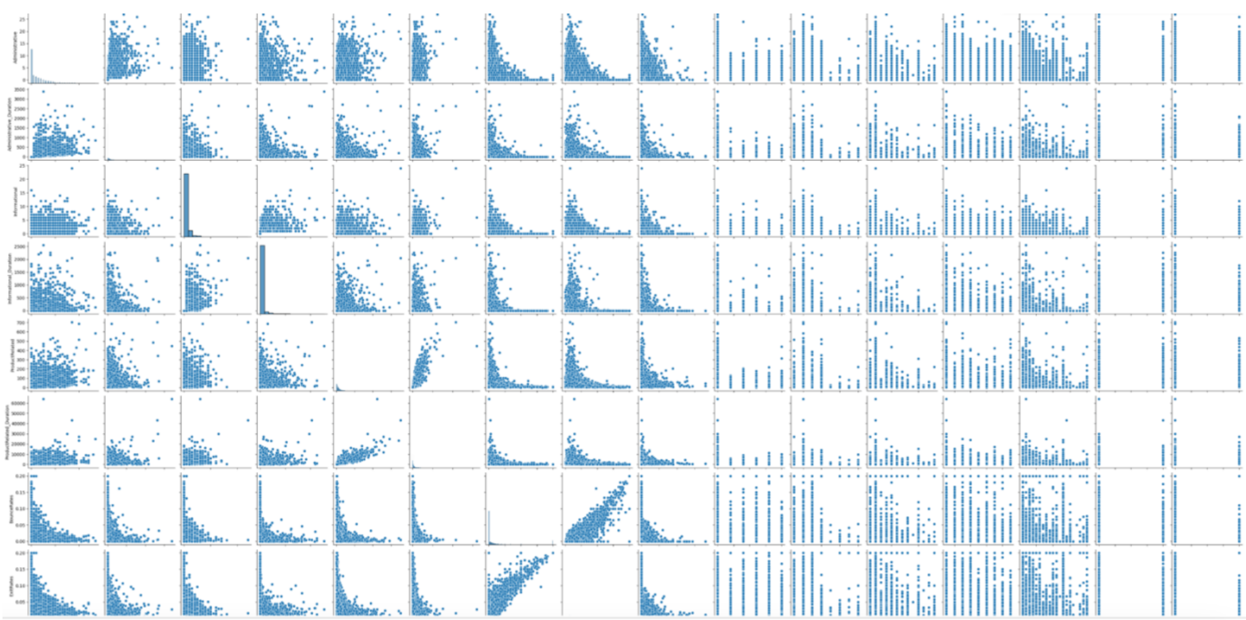
For each continuous variable, describe central and variational measures

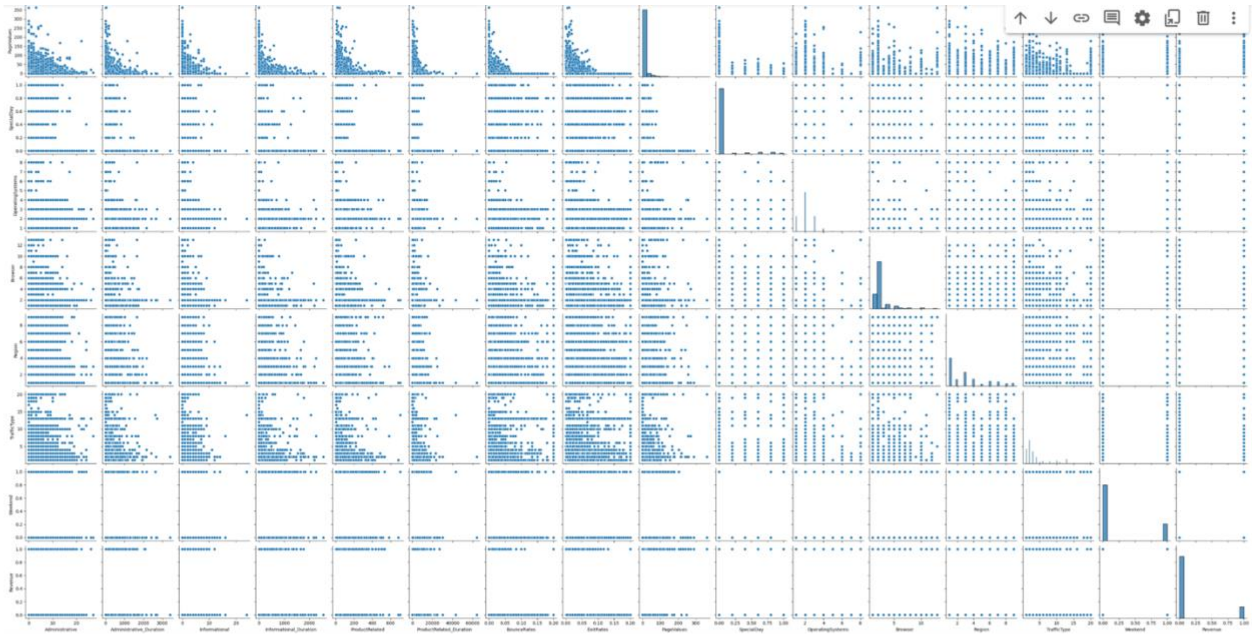
The table below describes the central and variational measures of the continuous variables. It is clear that each of the features contain outliers that are dramatically increasing the mean of each of the features, with the medians being well below the maximum values.

	Administrative_Duration	Informational_Duration	ProductRelated_Duration	BounceRates	ExitRates	PageValues
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000
mean	80.818611	34.472398	1194.746220	0.022191	0.043073	5.889258
std	176.779107	140.749294	1913.669288	0.048488	0.048597	18.568437
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	184.137500	0.000000	0.014286	0.000000
50%	7.500000	0.000000	598.936905	0.003112	0.025156	0.000000
75%	93.256250	0.000000	1464.157214	0.016813	0.050000	0.000000
max	3398.750000	2549.375000	63973.522230	0.200000	0.200000	361.763742

Produce appropriate graphs for the relationship between all of the features and calculate correlation coefficients

The below graphs were created using the pairplot() function from the seaborn library to show the relationships between all of the features.





The following tables display the calculated correlation coefficients. The 'TrafficType' feature has a weak correlation coefficient in relation to the 'Revenue' and is dropped.

	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
OperatingSystems	−0.006347	−0.007343
Browser	−0.025035	−0.015392
Region	−0.005487	−0.005561
TrafficType	−0.033561	−0.014376
Weekend	0.026417	0.014990
Revenue	0.138917	0.093587

	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
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
Weekend	0.016092	0.007311	−0.046514
Revenue	0.158538	0.152373	−0.150673

	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
OperatingSystems	−0.009527	−0.009579
Browser	−0.038235	−0.019285
Region	−0.029169	−0.027144
TrafficType	−0.034491	−0.024675
Weekend	0.035785	0.024078
Revenue	0.095200	0.070345

	ExitRates	PageValues	SpecialDay	OperatingSystems \
Administrative	−0.316483	0.098990	−0.094778	−0.006347
Administrative_Duration	−0.205798	0.067608	−0.073304	−0.007343
Informational	−0.163666	0.048632	−0.048219	−0.009527
Informational_Duration	−0.105276	0.030861	−0.030577	−0.009579
ProductRelated	−0.292526	0.056282	−0.023958	0.004290
ProductRelated_Duration	−0.251984	0.052823	−0.036380	0.002976
BounceRates	0.913004	−0.119386	0.072702	0.023823
ExitRates	1.000000	−0.174498	0.102242	0.014567
PageValues	−0.174498	1.000000	−0.063541	0.018508
SpecialDay	0.102242	−0.063541	1.000000	0.012652
OperatingSystems	0.014567	0.018508	0.012652	1.000000
Browser	−0.004442	0.045592	0.003499	0.223013
Region	−0.008907	0.011315	−0.016098	0.076775
TrafficType	0.078616	0.012532	0.052301	0.189154
Weekend	−0.062587	0.012002	−0.016767	0.000284
Revenue	−0.207071	0.492569	−0.082305	−0.014668

	Browser	Region	TrafficType	Weekend	Revenue
Administrative	−0.025035	−0.005487	−0.033561	0.026417	0.138917
Administrative_Duration	−0.015392	−0.005561	−0.014376	0.014990	0.093587
Informational	−0.038235	−0.029169	−0.034491	0.035785	0.095200
Informational_Duration	−0.019285	−0.027144	−0.024675	0.024078	0.070345
ProductRelated	−0.013146	−0.038122	−0.043064	0.016092	0.158538
ProductRelated_Duration	−0.007380	−0.033091	−0.036377	0.007311	0.152373
BounceRates	−0.015772	−0.006485	0.078286	−0.046514	−0.150673
ExitRates	−0.004442	−0.008907	0.078616	−0.062587	−0.207071
PageValues	0.045592	0.011315	0.012532	0.012002	0.492569
SpecialDay	0.003499	−0.016098	0.052301	−0.016767	−0.082305
OperatingSystems	0.223013	0.076775	0.189154	0.000284	−0.014668
Browser	1.000000	0.097393	0.111938	−0.040261	0.023984
Region	0.097393	1.000000	0.047520	−0.000691	−0.011595
TrafficType	0.111938	0.047520	1.000000	−0.002221	−0.005113
Weekend	−0.040261	−0.000691	−0.002221	1.000000	0.029295
Revenue	0.023984	−0.011595	−0.005113	0.029295	1.000000

Data Preprocessing

Split the dataset into 80% training and 20% testing and explain why this is necessary for ML.

It is necessary to split The Online Shoppers Purchasing Intention Dataset into 80% training and 20% testing for machine learning as the test data should be completely independent from the training data. The model that is developed is based on the training data and then tested on the test data to discover how well the trained model works on unseen data. If all the data was used in the training of the model, there would be no standalone data left to evaluate the trained model's performance.

```
# X variables include each feature except the 'Revenue' feature
X = df.drop(columns='Revenue')

# Y variable is the 'Revenue' feature
Y = df[['Revenue']]
```

```
# Import the train_test_split function from sklearn
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
# X and Y are assumed to be defined earlier in the code
# random_state parameter sets the seed for random number generation to ensure reproducibility
# shuffle parameter shuffles the data before splitting
# test_size parameter specifies the proportion of the dataset to include in the test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=23, shuffle=True, test_size=0.2)
```

Balance the dataset if necessary

With the dataset being imbalanced it is necessary for it to be balanced. To do this the Min-Max Scaler is imported from the sklearn.preprocessing library. The training data is then scaled by fitting the scaler to the data and transforming it. Once the features of the test data are scaled using the same scaler parameters learned from the training data the 'RandomOverSampler' clas is imported from the imbalanced-learn library. Using a random state of 42 an instance of

RandomOverSampler class is created and the distribution is then balanced. The ‘Revenue’ feature is then balanced as seen in the below table.

```
Revenue
False      8329
True       8329
```

Convert any categorical variables into dummy variables

Dummy variables are created for the ‘VisitorType,’ ‘SpecialDay,’ ‘Month,’ and ‘Weekend’ features.

```python
# Create an empty list to store dummy variables
dummies = []

# Define a list of columns for which dummy variables will be created
cols = ['SpecialDay', 'Month', 'VisitorType', 'Weekend']

# Iterate over each column in the list
for col in cols:
# Create dummy variables for the current column and append them to the list
    dummies.append(pd.get_dummies(df[col]))
```

The ‘SpecialDay,’ ‘Month,’ and ‘Weekend’ are then renamed.

	0.0	0.2	0.4	0.6	0.8	1.0	Aug	Dec	Feb	Jul	...	Mar	May	Nov	Oct	Sep	New_Visitor	Other	Returning_Visitor	False	True
0	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	1	0
1	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	1	0
2	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	1	0
3	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	1	0
4	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12325	1	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	1	0	1
12326	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	1	0	1
12327	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	1	0	1
12328	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	1	1	0
12329	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	1	0	0	0	1

12330 rows × 21 columns

```python
# Rename the new columns so they make more sense
df = df.rename(columns={0.0:'SpecialDay1',
                        0.2:'SpecialDay2',
                        0.4:'SpecialDay3',
                        0.6:'SpecialDay4',
                        0.8:'SpecialDay5',
                        1.0:'SpecialDay6',
                        'Feb':'Month_February',
                        'Mar':'Month_March',
                        'May':'Month_May',
                        'June':'Month_June',
                        'Jul':'Month_July',
                        'Aug':'Month_August',
                        'Sep':'Month_September',
                        'Oct':'Month_October',
                        'Nov':'Month_November',
                        'Dec':'Month_December',
                        False: 'Not_Weekend',
                        True: 'Weekend'})
```

Handle any missing data

As mentioned at the beginning of the report there are no null values present in the dataset. This is unchanged after the creation of the new features from the dummy variables, as seen in the table below.

```
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 34 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   OperatingSystems         12330 non-null  int64
 10  Browser                  12330 non-null  int64
 11  Region                   12330 non-null  int64
 12  Revenue                  12330 non-null  bool
 13  Not_Weekend              12330 non-null  uint8
 14  SpecialDay2              12330 non-null  uint8
 15  SpecialDay3              12330 non-null  uint8
 16  SpecialDay4              12330 non-null  uint8
 17  SpecialDay5              12330 non-null  uint8
 18  Weekend                  12330 non-null  uint8
 19  Month_August             12330 non-null  uint8
 20  Month_December           12330 non-null  uint8
 21  Month_February           12330 non-null  uint8
 22  Month_July               12330 non-null  uint8
 23  Month_June               12330 non-null  uint8
 24  Month_March              12330 non-null  uint8
 25  Month_May                12330 non-null  uint8
 26  Month_November           12330 non-null  uint8
 27  Month_October            12330 non-null  uint8
 28  Month_September          12330 non-null  uint8
 29  New_Visitor              12330 non-null  uint8
 30  Other                    12330 non-null  uint8
 31  Returning_Visitor        12330 non-null  uint8
 32  Not_Weekend              12330 non-null  uint8
 33  Weekend                  12330 non-null  uint8
dtypes: bool(1), float64(6), int64(6), uint8(21)
memory usage: 1.4 MB
```

Outliers

Explain chebyshev's rule and use it to find any outliers in the data

According to Monhor & Takemoto (2005), Chebyshev's rule is tool most commonly used for proving different convergence processes. They describe it as playing a "fundamental role in proofs of various forms of laws of large numbers." For a random variable X with a mean μ and a standard deviation σ, Chebyshev's Rule can be expressed as:

$$P(|X - \mu| \geq k\sigma) \leq 1/k^2$$

The rule gives a bound for the probability of deviation of a random variable from its mathematical expectation in terms of its variance. Assigning k the value of 2, assumes that 75% of the data should fall within 2 standard deviations of the mean.

```
Outliers:
       Administrative Informational ProductRelated OperatingSystems Browser
62                 12           NaN            NaN              NaN     NaN
76                 10           NaN            NaN              NaN     NaN
188                 9           NaN            NaN              NaN     NaN
248                16           NaN            NaN              NaN     NaN
282                13           NaN            NaN              NaN     NaN
...               ...           ...            ...              ...     ...
12234             NaN           NaN            NaN              NaN     NaN
12247             NaN           NaN            NaN              NaN     NaN
12272             NaN           NaN            NaN              NaN     NaN
12287             NaN           NaN            NaN              NaN     NaN
12313             NaN           NaN            NaN              NaN     NaN

       Region  Administrative_Duration  Informational_Duration  \
62        NaN                      NaN                     NaN
76        NaN                      NaN                     NaN
188       NaN                      NaN                     NaN
248       NaN                      NaN                     NaN
282       NaN                      NaN                     NaN
...       ...                      ...                     ...
12234     NaN                      NaN                     NaN
12247     NaN                      NaN                     NaN
12272     NaN                      NaN                     NaN
12287     NaN                      NaN                     NaN
12313     NaN                      NaN                     NaN

        ProductRelated_Duration  BounceRates  ExitRates  PageValues
62                          NaN          NaN        NaN         NaN
76                          NaN          NaN        NaN         NaN
188                         NaN          NaN        NaN         NaN
248                         NaN          NaN        NaN         NaN
282                         NaN          NaN        NaN         NaN
...                         ...          ...        ...         ...
12234                       NaN          NaN        NaN   81.027296
12247                       NaN          NaN        NaN  133.281379
12272                       NaN          NaN        NaN   97.860836
12287                       NaN          NaN        NaN   44.219794
12313                       NaN          NaN        NaN   78.811725

[2025 rows x 12 columns]
```

Explain the box plot technique and use it to find any outliers in the data

The box plot technique is way of graphically representing a dataset's distribution. It includes the first quartile (Q1), median (second quartile or Q2), third quartile (Q3), and maximum values. The box in the box plot represents the interquartile range (IQR), spanning from Q1 to Q3. The length of the box shows the spread of the middle 50% of the data. The central tendency and median (Q2) is represented by the line inside the box. The whiskers extend from the box to the minimum and maximum values within a range of 1.5 times the IQR. Values beyond the whiskers are considered outliers.

In The Online Shoppers Purchasing Intention Dataset (Sakar & Kastro, 2018), there are 0 outliers within the lower bounds of either of the 6 continuous features. However, in the upper bounds of each continuous feature there are many outliers. In the upper bounds the 'Administrative_Duration' feature contains 935 outliers, the 'Informational_Duration' feature contains 1,923 outliers, the 'ProductRelated_Duration' contains 787 outliers, the 'BounceRates' contains 1,238 outliers, the 'ExitRates' contains 868 outliers, and the 'PageValues' contains 2,225 outliers. An example of the code using the 'Administrative_Duration' feature can be seen below.

```
# Applying the Box Plot Outlier Technique to the Administrative_Duration column
# Importing the necessary function from the scipy library
from scipy.linalg.special_matrices import dft

# Calculating the 75th and 25th percentiles (Q3 and Q1 respectively) of the Administrative_Duration column in the training data (X_train)
q3, q1 = X_train.Administrative_Duration.quantile(0.75), X_train.Administrative_Duration.quantile(0.25)

# Printing the values of Q3 and Q1
print('Q3:', q3, '| Q1:', q1)

# Calculating the interquartile range (IQR)
IQR = q3 - q1

# Printing the value of the interquartile range (IQR)
print('IQR:', IQR)

# Calculating the upper and lower bounds for detecting outliers using the box plot technique
upper_bound = q3 + 1.5 * IQR
lower_bound = q1 - 1.5 * IQR

# Printing the upper and lower bounds
print('Upper Bound:', upper_bound)
print('Lower Bound:', lower_bound)
```

```
Q3: 94.0 | Q1: 0.0
IQR: 94.0
Upper Bound: 235.0
Lower Bound: -141.0
```

Discuss any outliers and decide if you will remove / alter these values and why

With the box plot technique being applied to each of the continuous features it is evident that there are 5,235 rows out of the 9,864 rows in the X_train effected by outliers. With 53% of the rows containing outliers it is likely that these rows will impact the results of the logistic regression model. While removing 53% of the data is a substantial amount of data to remove there is still enough data to train the model on with there still being 4,629 instances to train the model on. Due to this, each row that contains an outlier is removed from the X_train DataFrame to improve the trained model's performance. The code used as well as the X_train without the outliers can be seen below.

```
# Importing the necessary function from the scipy library
from scipy.linalg import dft

# Features to handle outliers
features = ['Administrative_Duration', 'Informational_Duration', 'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues']

# Loop through each feature to handle outliers
for feature in features:
    # Calculating the 75th and 25th percentiles (Q3 and Q1 respectively) of the feature column in the training data (X_train)
    q3, q1 = X_train[feature].quantile(0.75), X_train[feature].quantile(0.25)

    # Calculating the interquartile range (IQR)
    IQR = q3 - q1

    # Calculating the upper and lower bounds for detecting outliers using the box plot technique
    upper_bound = q3 + 1.5 * IQR
    lower_bound = q1 - 1.5 * IQR

    # Filtering rows with outliers and removing them from the training data (X_train)
    X_train = X_train[(X_train[feature] >= lower_bound) & (X_train[feature] <= upper_bound)]

# Now X_train contains only rows without outliers in any of the specified features
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates
7596	3	133.866667	0	0.0	7	128.000000	0.00000	0.026667
2591	0	0.000000	0	0.0	15	455.500000	0.00000	0.014286
9700	0	0.000000	0	0.0	2	93.000000	0.00000	0.050000
7993	0	0.000000	0	0.0	20	821.833333	0.05000	0.076667
11453	0	0.000000	0	0.0	32	1109.083333	0.00125	0.019375
...	...	...	...	...	...	...	...	...
3674	6	93.500000	0	0.0	23	251.333333	0.00000	0.013768
39	0	0.000000	0	0.0	9	482.000000	0.00000	0.022222
11190	1	33.500000	0	0.0	33	880.633333	0.00000	0.034118
10185	0	0.000000	0	0.0	15	729.000000	0.00000	0.033333
9256	0	0.000000	0	0.0	10	426.500000	0.00000	0.020000

4629 rows × 33 columns

Explain the rationale for normalization, and use some appropriate technique to normalize your data.

According to Dublin Business School (2024), normalisation brings all the values of numeric columns in the dataset to a common scale. Normalisation involves either scaling the features to have a mean of 0 and a standard deviation of 1 or scaling the features to a range between 0 and 1, the first technique being standardisation and the second technique being min-max scaling. Both techniques ensure that all features have similar ranges and variances so that any potential issues are addressed.

Min-max scaling was the chosen method of normalisation of The Online Shoppers Purchasing Intention Dataset (Sakar & Kastro, 2018). This is because no feature is on a normal scale. Each feature has a very different range, so by using the min-max scaler to normalise the values, the features will have similar ranges as they will all range from 0 to 1.

```
from sklearn.preprocessing import MinMaxScaler
# Create an instance of the MinMaxScaler class
scaler = MinMaxScaler()

# Scale the features of the training data (X_train) to a specified range (typically between 0 and 1)
# by fitting the scaler to the data and transforming it
X_train_scaled = scaler.fit_transform(X_train)

# Scale the features of the test data (X_test) using the same scaler parameters learned from the training data
X_test_scaled = scaler.transform(X_test)
```

Predictive Analysis

Specify input and output variables

The input variables are all features except for 'Revenue'. The output variable is the 'Revenue' feature.

```
# Specify input features (All features less the 'Revenue' feature)
X = df.drop(columns='Revenue')

# Specify output feature
Y = df[['Revenue']]
```

Fit a suitable regression model to your training dataset

A logistic regression model is chosen as the regression model to fit to the dataset. Logistic regression models are suitable for data that has an output carriable containing categorical data. The 'Revenue' feature contains Boolean data, with 0 indicating the customer not making a purchase and 1 indicating the customer made a purchase, making the feature categorical, therefore a logistic regression model is a suitable model to fit to the training dataset. The code used to fit the logistic regression model uses the LogisticRegression module from the sklearn.linear_model library and can be seen below. The model has a 70.68% accuracy on test data and an 80.07% accuracy on the resampled training data.

```
# Import the LogisticRegression class from scikit-learn
clf = LogisticRegression()

# Fit the logistic regression model to the resampled training data
clf.fit(X_train_scaled_resampled,Y_train_resampled)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. I
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()

# Make predictions on the scaled test data
Y_pred = clf.predict(X_test_scaled)

# Calculate the accuracy score of the model on the test data
clf.score(X_test_scaled,Y_test)

0.7068126520681265

# Calculate the accuracy score of the model on the resampled training data
clf.score(X_train_scaled_resampled,Y_train_resampled)

0.8006963621082963
```

Interpret the model parameters

The logistic regression model is created using the Logit function from statsmodels.It is then fitted to the training data and a summary of the fitted logistic regression model is printed. When analysing the p>|z| values to look for features with values less than 0.05 it is clear that the 'ExitRates,' 'PageValues,' 'OperatingSystems,' and 'Browser' features are the features that are the stronger predictors in the model. The other features are dropped and the model is refitted using the aforementioned features. The summary of this refitted model can be seen below.

```
Optimization terminated successfully.
         Current function value: 0.317324
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                 9864
Model:                          Logit   Df Residuals:                     9860
Method:                           MLE   Df Model:                            3
Date:                Fri, 22 Mar 2024   Pseudo R-squ.:                  0.2660
Time:                        13:42:08   Log-Likelihood:                -3130.1
converged:                       True   LL-Null:                       -4264.5
Covariance Type:            nonrobust   LLR p-value:                     0.000
====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
ExitRates          -35.5101      1.909    -18.602      0.000     -39.252     -31.769
PageValues           0.0749      0.003     29.410      0.000       0.070       0.080
OperatingSystems    -0.5428      0.032    -17.085      0.000      -0.605      -0.481
Browser             -0.0602      0.021     -2.895      0.004      -0.101      -0.019
====================================================================================
```

Use your model to predict the output for your test data

The logistic regression model is fitted on the training data and a score is formed using the test data to evaluate its performance. The model gets a score of 88.36% on the test data.

```
# Fit the model on the training data
clf.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. 
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
# Predict on the test data
Y_pred = clf.predict(X_test)
```

```
# Evaluate performance on the test data
clf.score(X_test,Y_test)
```

```
0.883617193836172
```

Provide an analysis of the model performance

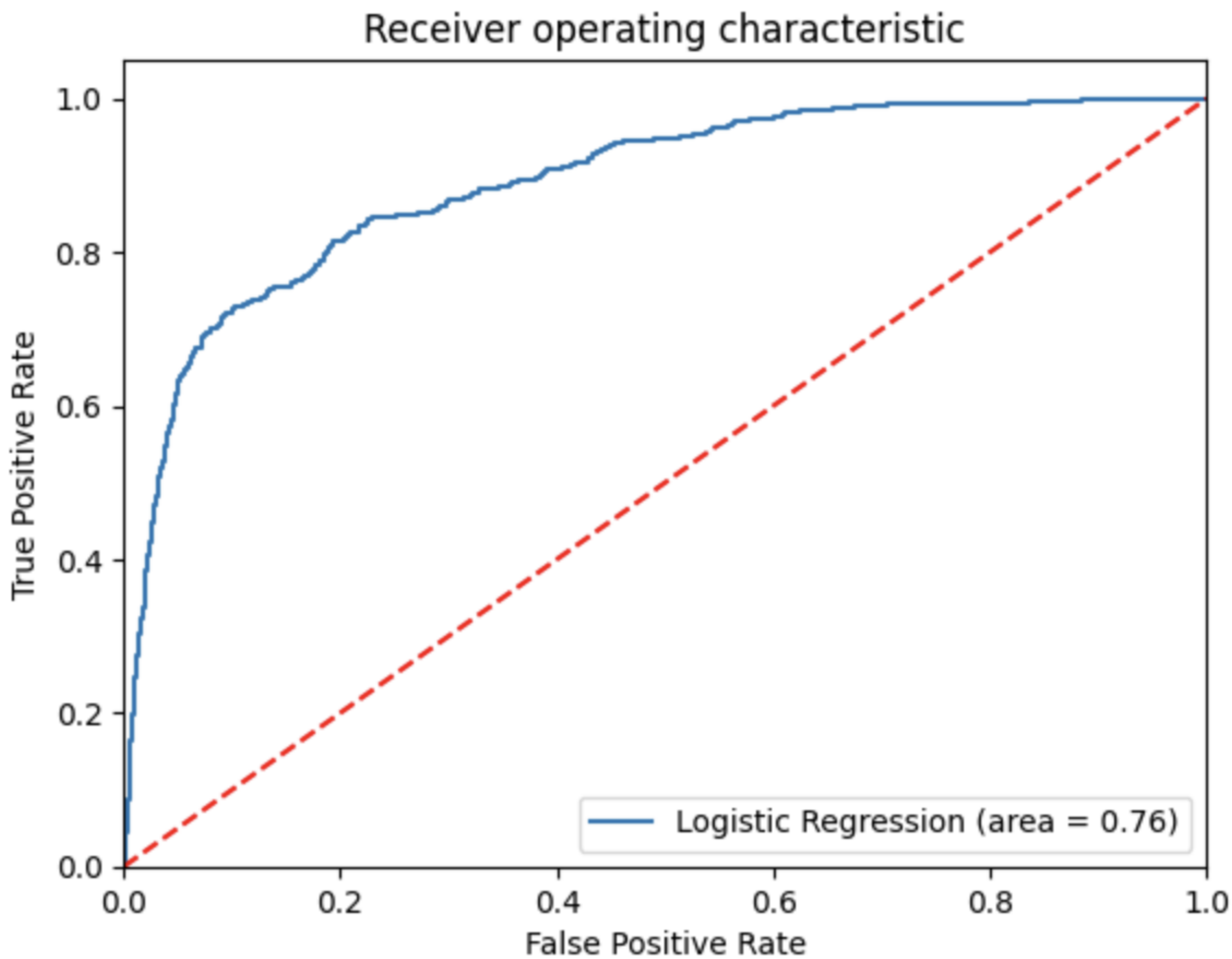
From the confusion matrix below describes the accuracy of the model. The previously mentioned accuracy score of 88.36% originates from the model's true positives being 143, its true negatives being 2,036, its false positives being 57 and its false negatives being 230.

```
[[2036   57]
 [ 230  143]]
```

The precision, recall, and F1 score give a more comprehensive understanding of the model. The precision score of 71% measures the proportion of true positive predictions among all positive predictions made. This indicates that when the model predicts a positive outcome, it is correct about 71.5% of the time. The recall score of 38% measures the proportion of true positive cases that were correctly identified. This indicates that your model captures about 38.3% of all actual positive cases. The F1 score of 50% is the collective mean of precision and recall, providing a balance between the two metrics. The specificity score of 97% measures the proportion of true negative cases that were correctly identified. This indicates that among all the actual negative instances, the model correctly identifies around 97% of them as negative, and only around 2.8% of actual negative instances are incorrectly classified as positive.

	precision	recall	f1-score	support
False	0.90	0.97	0.93	2093
True	0.71	0.38	0.50	373
accuracy			0.88	2466
macro avg	0.81	0.68	0.72	2466
weighted avg	0.87	0.88	0.87	2466

The ROC Curve plots the sensitivity against the false positive rate. The ROC Curve below has an area under the curve of 0.76. This suggests that the model has a 76% chance of distinguishing between positive and negative classes. An area under the curve of 0.76 indicates that the logistic regression model performs better than random guessing but still needs improvement.



Conclusion

In conclusion, the fitted logistic regression demonstrates an overall good performance across various metrics. The accuracy of 88.36% indicates that it effectively classifies cases into their respective categories. Although the precision stands at 71.5%, indicating its ability to correctly identify positive instances, a recall score of 38.3% suggests room for improvement in capturing all positive cases. The model's high specificity score of 97% demonstrates its proficiency in correctly identifying negative instances. Additionally, with an F1 score of 0.5 and an AUC of 0.76, the model exhibits reasonable discriminative ability. Overall, while the model shows promise, further optimization may be beneficial for enhancing its predictive capabilities.

References

Dublin Business School. (2024) ‘Classification - Logistic Regression’, B8IT160: Applied Data Analytics. [Lecture Slides]. Available at: https://elearning.dbs.ie/pluginfile.php/2104676/mod_resource/content/1/Classification%20-%20Logistic%20Regression.pdf (Accessed: 20th March 2024).

Dublin Business School. (2024) ‘Notebook: Logistic Regression - Titanic Data’, B8IT160: Applied Data Analytics. [Google Colab]. Available at: https://colab.research.google.com/drive/1P1q3VFeZagJ2cUDrZ82IUulo1SzdRCxn?usp=sharing#scrollTo=-hjCi7KBZWHF (Accessed: 10th March 2024).

Monhor, D., Takemoto, S. (2005) Understanding the concept of outlier and its relevance to the assessment of data quality: Probabilistic background theory. *Earth Planet Sp* 57, 1009–1018. Available at: https://doi.org/10.1186/BF03351881 (Accessed: 15th March 2024).

Sakar, C., & Kastro, Y. (2018) *Online Shoppers Purchasing Intention Dataset* [Dataset]. Available at: https://doi.org/10.24432/C5F88Q (Accessed 5th March 2024).