IE7300 FINAL PROJECT

Online Shoppers Purchase Classification

Group 6

Prayansh Maheshwari

Neel Anap

maheshwari.pray@northeastern.edu

anap.n@northeastern.edu

Percentage of Effort Contributed by Student 1: _50%_

Percentage of Effort Contributed by Student 2: _50%_

Signature of Student 1: *Prayansh Maheshwari*

Signature of Student 2: *Neel Anap*

Submission Date : 12/04/22

1. Problem Setting

Marketing analytics is the study of data to evaluate the effectiveness of marketing initiatives. Businesses can improve their marketing campaigns, comprehend what motivates consumer behavior, and maximize their return on investment. In this project, we'll examine whether a customer bought a product as a result of some website features. From a marketing perspective, it would be very advantageous to analyze this trend and forecast product sales because it would assist the business in making various decisions, such as deciding whether to stock the product or plan for discounts.

2. Project Goal

A specific business that maintains a website to sell its goods in a virtual marketplace wants to forecast when a product will be bought. Finding patterns and similarities between customers who have purchased the product and those who have not will help us solve this issue by grouping the two groups of customers into clusters. Numerous elements or characteristics that the business has learned about from its website ultimately affect the customer's choice to buy the product. These are the variables that we intend to use to predict and build a dependency on a purchase.

3. Data Description

The dataset for this project is taken from the UCI Machine Learning Repository.
Link - UCI Machine Learning Repository: Online Shopper's Purchase Intention
The dataset containing the site's statistics is called "Online Shoppers' Purchase Intention." The dataset has 18 columns and approximately 12K rows. There are 10 numerical and 8 categorical variables. The target variable is a string variable named "Purchased," which marks whether the customer purchased a product or not.

The attribute description is as follows:

Column Name	**Description**
Administrative	Represents ‘Administrative’ type pages visited by customers in a session
Administrative_Duration	Represents time spent on ‘Administrative’ type pages visited by customers in a session

Informational	Represents ‘Informational’ type pages visited by customers in a session
Informational_Duration	Represents time spent on ‘Informational’ type pages visited by customers in a session
ProductRelated	Represents ‘ProductRelated’ type pages visited by customers in a session
ProductRelated_Duration	Represents time spent on ‘ProductRelated’ type pages visited by customers in a session
BounceRates	The value of "Bounce Rate" feature for a web page refers to the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session
ExitRates	The value of "Exit Rate" feature for a specific web page is calculated as for all pageviews to the page, the percentage that were the last in the session
PageValues	The "Page Value" feature represents the average value for a web page that a user visited before completing an e-commerce transaction
SpecialDay	The "Special Day" feature indicates the closeness of the site visiting time to a specific special day (e.g., Mother’s Day, Valentine's Day) in which the sessions are more likely to be finalized with a transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date
Month	Month in which the session occurred
OperatingSystems	The operating system used in a particular session
Browser	The type of browser used in a particular session
Region	Numerical - representing different number for each region
TrafficType	Numerical - representing different number for each traffic type
VisitorType	Categorical - New or Returning user
Weekend	Weekend - True, Non-Weekend - False

Revenue	The target variable representing if the customer made a purchase or not

4. Data Exploration and Visualization

To make sure the dataset is complete, the null entries are examined first. Our dataset is found to have no missing or null values, making it suitable for use in machine learning tasks.

```
# Checking for null values
df.isnull().sum()
```

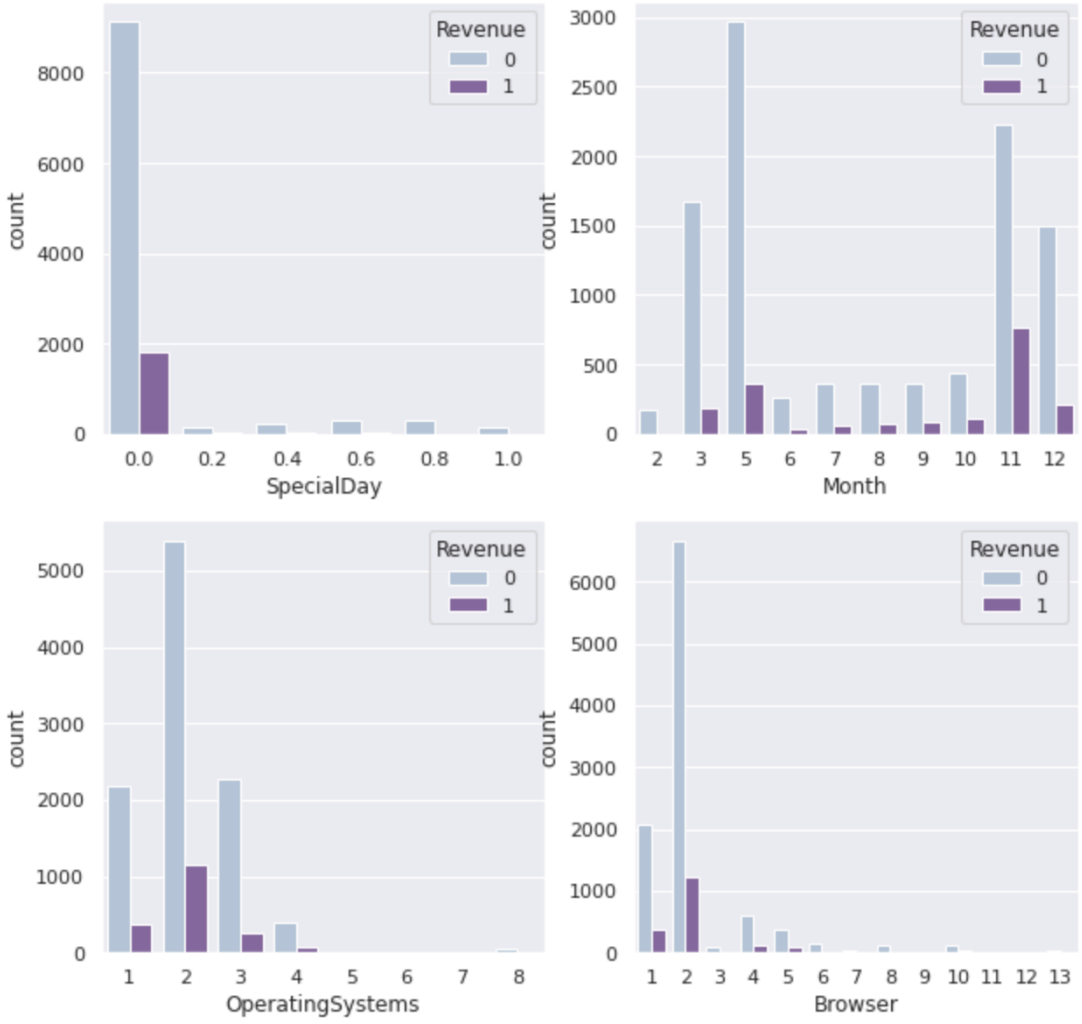
```
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

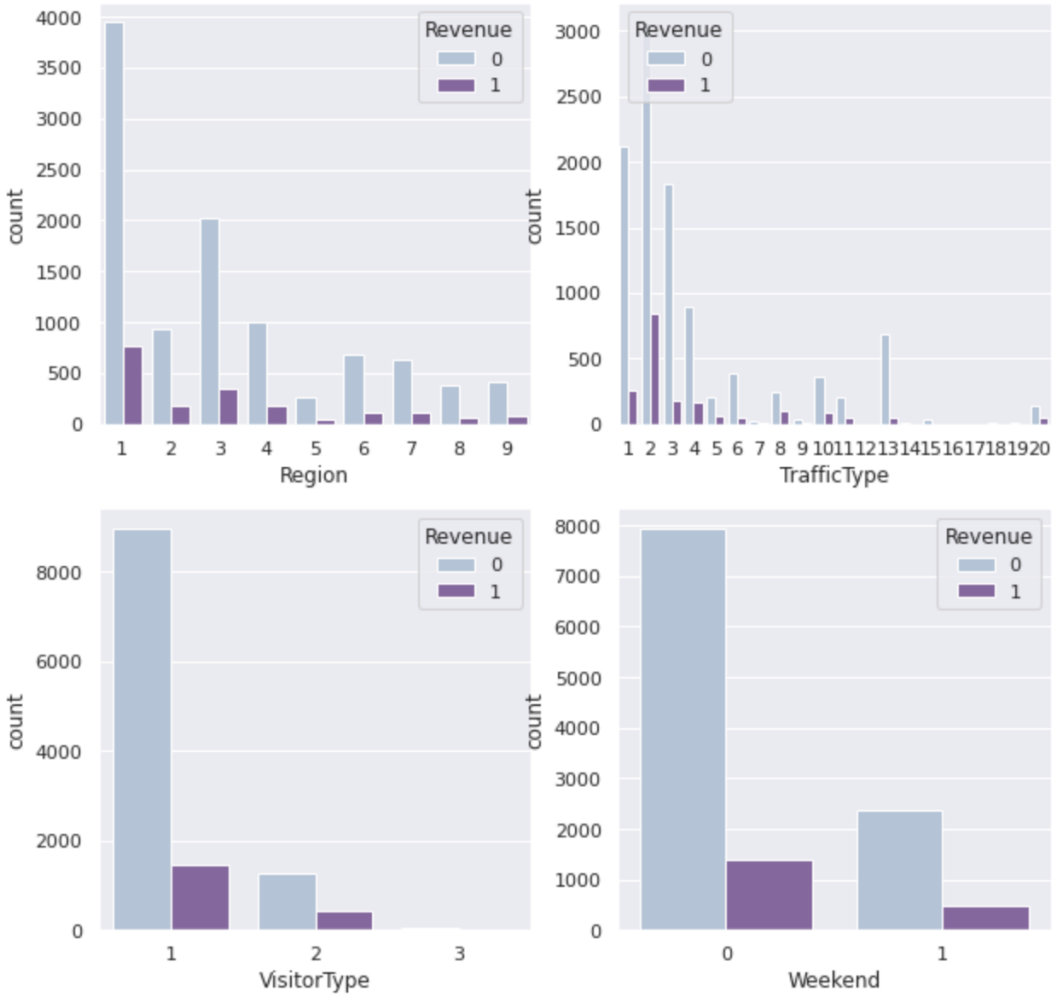
Then we checked the number of duplicate rows and removed them from the dataset. After this we moved ahead with exploring attributes.

4.1 Categorical Variables

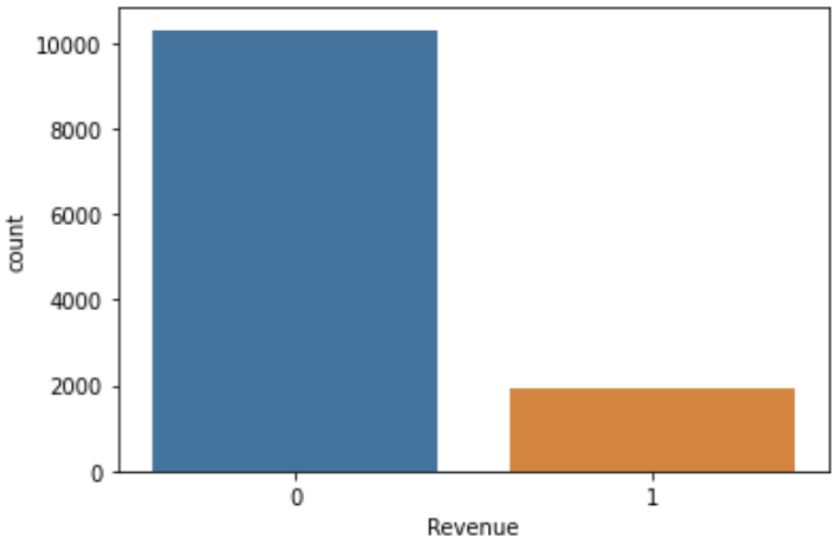
Distribution is presented for categorical variables with respect to the target variable "Revenue" (Product bought or not). Each categorical variable's impact on the customer's choice is shown. The graphs below show that the Special Day category has no bearing on the target variable. Looking at the months, we can observe that May, November, and December have larger sales of products than the other months, which may be related to the holiday

shopping season. Users only utilize a small number of operating systems and browsers, and as a result, a small number of them display the target variable response. The charts below show that regions and traffic categories 1 to 4 have the highest likelihood of making a purchase. Compared to other visitor types, visitor type 1 has the highest conversion potential. The graph shows that weekday users of the site had the high probability of making a purchase, indicating that the weekend does not have a significant impact on customer purchases.





We can see the percentage of clients who buy or don't buy a product from the plot below. Our data is very imbalanced, with more than 80% being 0 and 20% being 1.

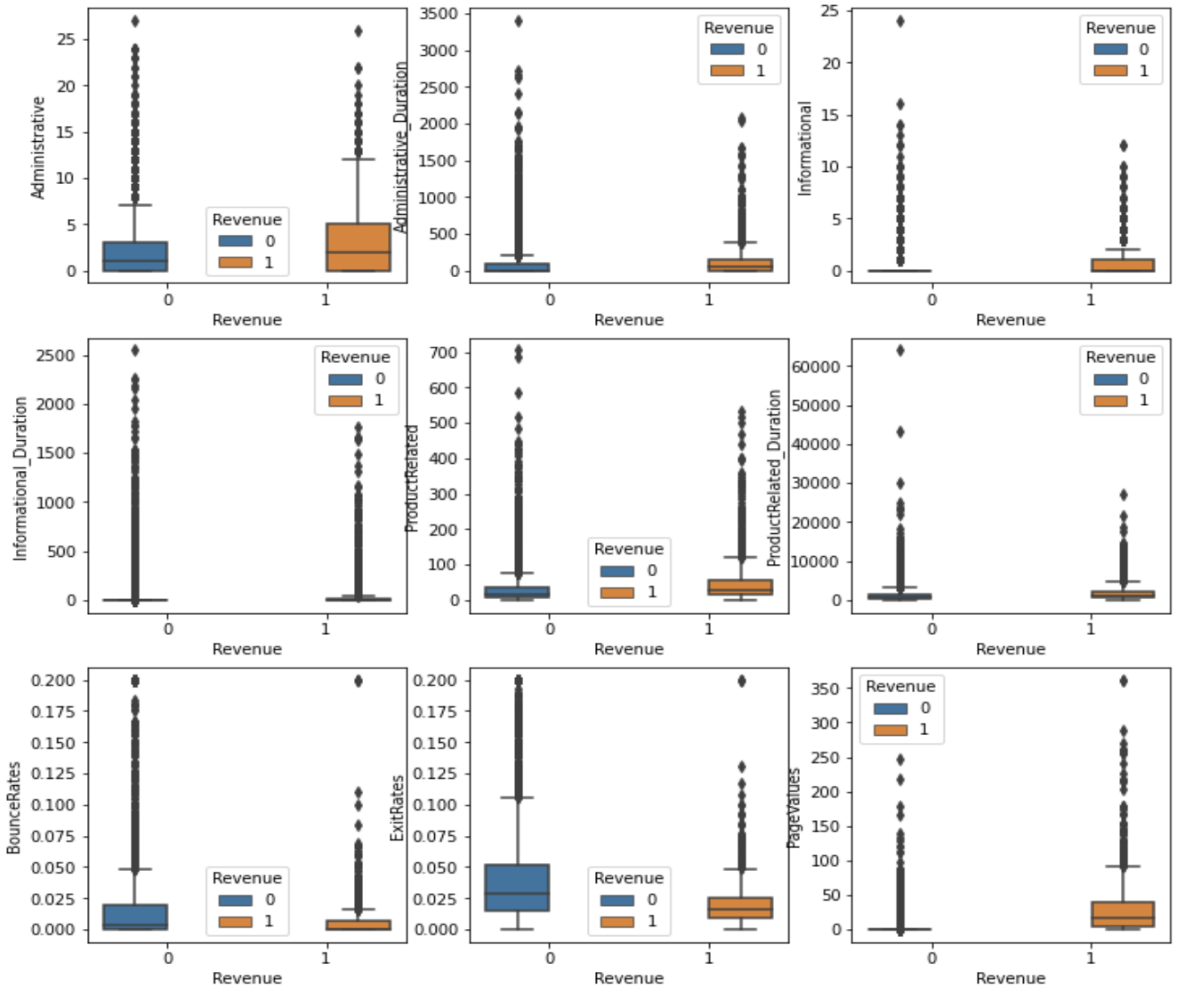


4.2 Numerical Variables

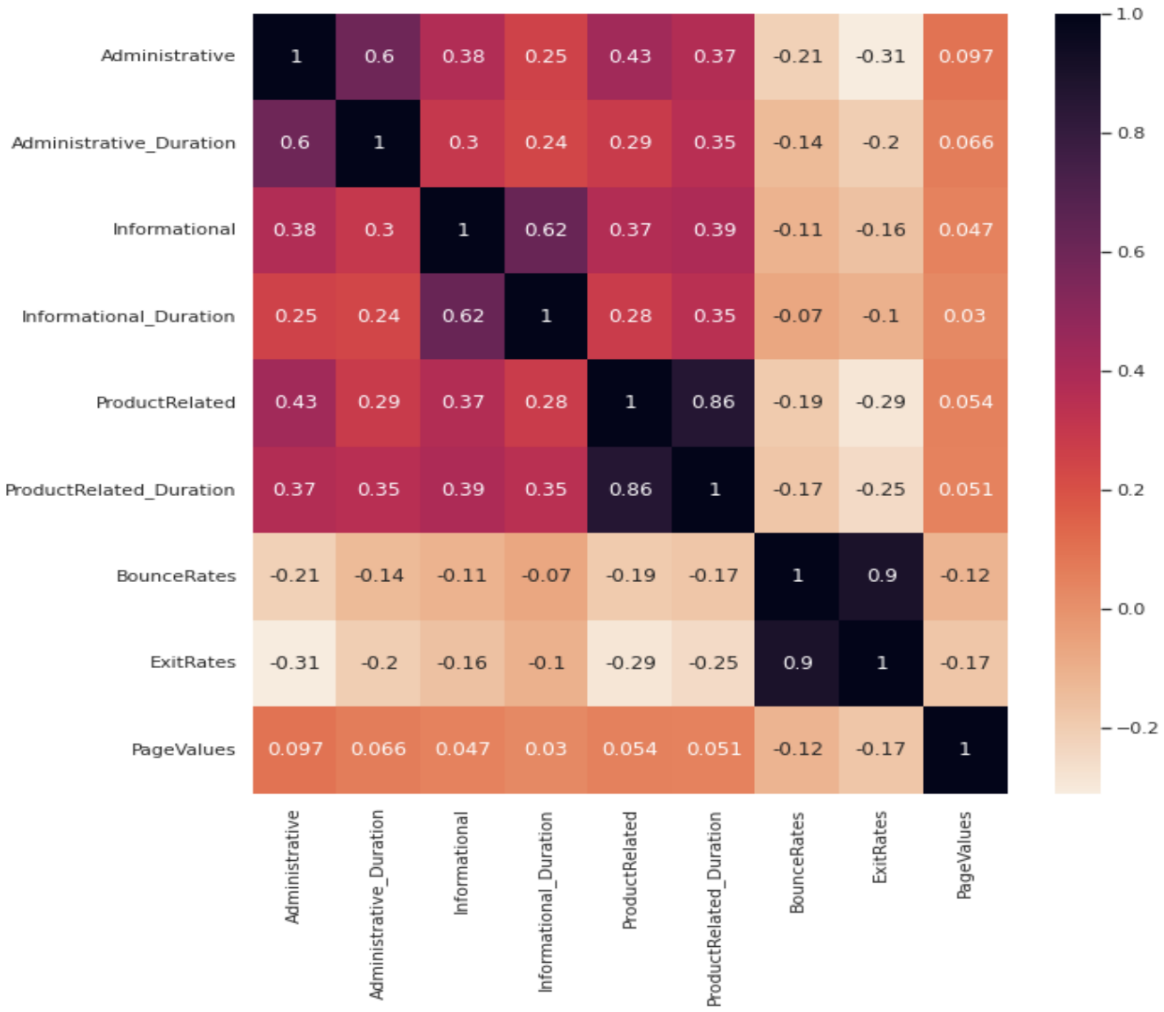
Summary of the numerical variables. We can see that most of the variables are right skewed.

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues
count	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000	12205.000000
mean	2.338878	81.646331	0.508726	34.825454	32.045637	1206.982457	0.020370	0.041466	5.949574
std	3.330436	177.491845	1.275617	141.424807	44.593649	1919.601400	0.045255	0.046163	18.653671
min	0.000000	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	0.000000
50%	1.000000	9.000000	0.000000	0.000000	18.000000	608.942857	0.002899	0.025000	0.000000
75%	4.000000	94.700000	0.000000	0.000000	38.000000	1477.154762	0.016667	0.048529	0.000000
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	0.200000	361.763742

Similar to how most real-world data is right skewed, boxplots are also produced for numerical variables, showing that our data lacks a distinct center.

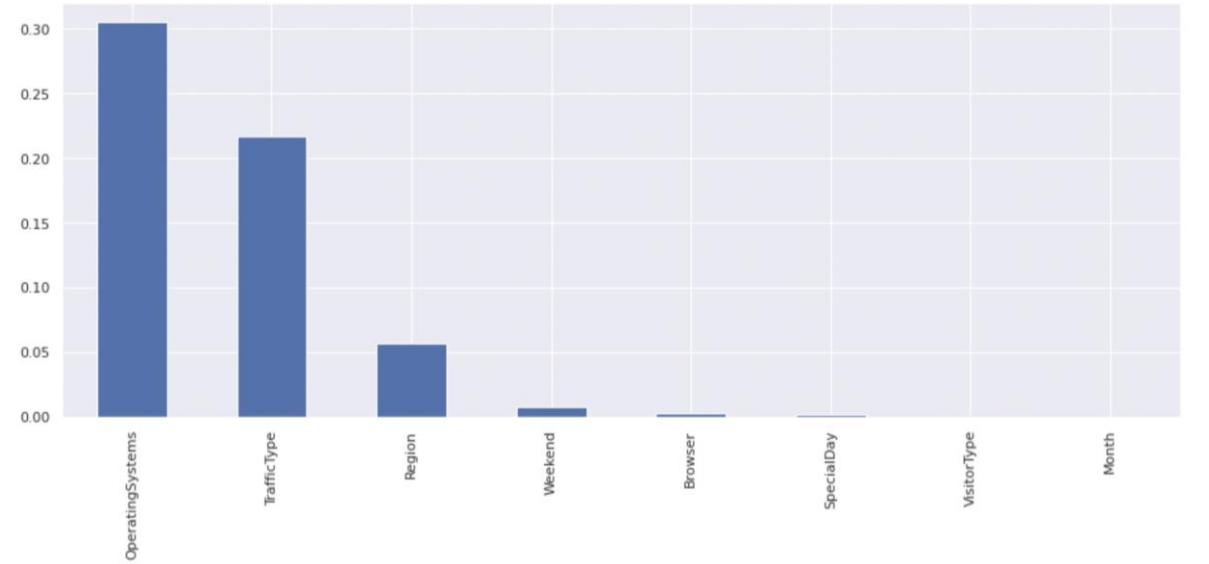


To determine the correlation between the numerical variables, we have also created a correlation heatmap. We can see high correlation between Administrative and Administrative_duration, Informational and Informational_duration, ProductRelated and ProductRelated_duration, and BounceRates and ExitRates. And few columns showing negative correlation as well.



5. Data Pre-processing

For the purpose of choosing features for categorical variables, the chi square test has been used. First, the categorical data was transformed into nominal categories before the chi test was performed. From the graph below we can he variables OperatingSystems and TrafficType has a p value larger than 0.05, hence these variables are dropped from the dataset.



Following that, we examined the skewness of the numerical variables. The majority of the variables are clearly severely skewed, as can be seen in the table below. To fix this skewness in the dataset, we applied the square root technique.

Skewness before

Administrative	1.947123
Informational	4.014173
ProductRelated	4.333419
BounceRates	3.162425
PageValues	6.350983
Administrative_Duration	5.592152
Informational_Duration	7.540291
ProductRelated_Duration	7.253161
ExitRates	2.234645

Skewness after

Administrative	0.625303
Administrative_Duration	1.527806
Informational	1.933381
Informational_Duration	3.419129
ProductRelated	1.503354
ProductRelated_Duration	1.412977
BounceRates	1.715256
ExitRates	1.211128
PageValues	2.515993

Following that, we removed the highly correlated columns from the dataset based on the correlation plot. The categorical attributes were then subjected to One hot encoding, resulting in the addition total of 47 columns. The data was divided into training (75%) and test (25%) sets. After this, as our dataset was imbalanced, we performed SMOTE technique of oversampling the minority class which in terms of our dataset was Revenue as 1. Then, before employing any machine learning models, the training data was standardized. The test data was then converted using the training data mean and standard deviation.

6. Machine Learning Models

The following models have been implemented:

1. Logistic Regression
2. Naïve Bayes
3. Neural Networks

6.1 Logistic regression

Logistic Regression is the most common classification model which uses the assumptions that the outcome is categorical in nature and that the predictors are not linearly coupled, which our data are. Logistic regression classifiers may be built from very small datasets, and once the model has been estimated, categorizing huge samples of new records is computationally straightforward and quick. We performed a cut off vs. sensitivity/accuracy study after making predictions and fitting the model to the training set.

Following the selection of the cut-off, we performed predictions on the test set, constructed a confusion matrix, and used the default cut-off.

This model has been implemented using the gradient descent algorithm. An iterative optimization process called gradient descent finds the minimum of a differentiable function. Throughout this process, we test different variables and make updates to them until we find the ones that minimizes the output.

How frequently we update the parameters is determined by the learning rate. If the learning rate is too high, we could "overshoot" the ideal value. The same is true if it is too small; it will take too long for us to attain the optimal levels. A vast range of input values have been mapped into a small range using the sigmoid function. Mathematically, sigmoid function is,

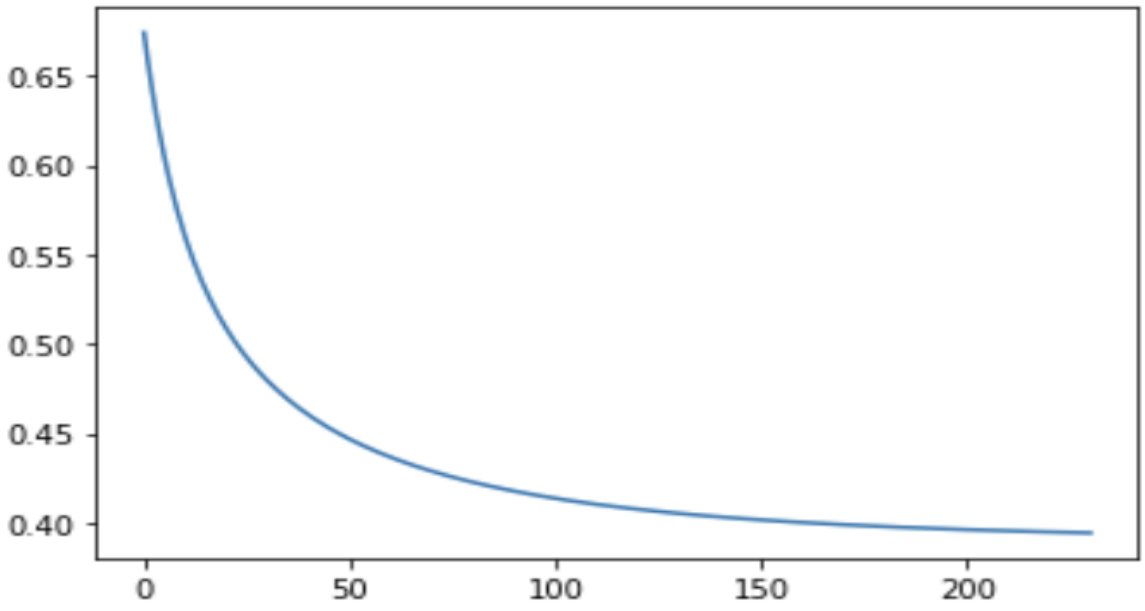
$$y = g(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

The model's performance is determined by the cost function. It is employed to gauge how closely the model's predictions match the actual results. The cost can be computed numerically as follows for specific “m” observations:

$$J(\theta) = \frac{1}{m}[\sum_{i=1}^{m} -y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))]$$

Then, we used the following parameters to train our model: Maximum iterations are 5000, Tolerance is 0.00005, and Alpha is 0.00001.

Below is the graph that was produced by the gradient descent algorithm. As the number of iterations rises, the error is depicted on the graph.



We estimated the result for training error using the weights, and we calculated cutoff vs sensitivity/accuracy to select the best cutoff. These are the outcomes:

```
lr.predict_train(stdtrainx, train_y)
```

```
[[5055 2696]
 [  81 7670]]
[[6152 1599]
 [ 256 7495]]
[[7091  660]
 [ 790 6961]]
[[7434  317]
 [1572 6179]]
[[7584  167]
 [2682 5069]]
[[7702   49]
 [4746 3005]]
   cutoff  sensitivity  accuracy
0    0.10     0.989550  0.820862
1    0.20     0.966972  0.880338
2    0.40     0.898078  0.906464
3    0.60     0.797187  0.878145
4    0.80     0.653980  0.816217
5    0.95     0.387692  0.690685
```

In order to achieve a better balance between sensitivity and accuracy, we choose to utilize a cutoff of 0.4. The following outcomes are obtained after applying our trained model to our testset:

```
lr.predict_test(stdtestx,test_y)
```

```
[[2338  208]
 [ 169  337]]
The test accuracy score of the model is 0.8764744429882044
The test sensitivity score of the model is 0.66600790513834
```

6.2 Gaussian Naïve Bayes

The continuation of Naïve Bayes is the Gaussian Naïve Bayes. The Gaussian or normal distribution is the simplest to implement because it only requires calculating the mean and standard deviation for the training data. It is employed when we assume that every continuous variable related to each feature is distributed uniformly.

The following is the Bayes Theorem calculation in its simplest version. Where the marginal probability of the occurrence P(A) is referred to as the prior and the probability that we are interested in computing P(A|B) is called the posterior probability.

P(A|B) = P(B|A) * P(A) / P(B)

The likelihood of a predictor for a given class is its probability. The number of examples (rows) in the training dataset divided by the proportion of observations with the class label can be used to determine the prior.

P(yi) = examples with yi / total example

The probability that many possible experiment outcomes will occur is provided by the probability distribution. By establishing assumptions regarding the distribution of the underlying observations or parameters, it enables you to build statistical models.

To determine if the model is overfitting or not, we trained the priors for the training set and produced predictions for both the training and test sets.

Training Dataset

```
gnb.predict(stdtrainx.values,train_y.values)
```

```
[[3324 4427]
 [1213 6538]]
6538
Sensitivity: 0.843504063991743 /n Accuracy: 0.6361759772932525
       y  y_hat
0      0      1
1      0      1
2      1      0
3      0      1
4      0      1
...   ..    ...
15497  1      1
15498  1      1
15499  1      1
15500  1      1
15501  1      1
```

Training Dataset

```
gnb.predict(stdtestx.values,test_y)
```

```
[[1086 1460]
 [ 229  277]]
277
Sensitivity: 0.5474308300395256 /n Accuracy: 0.44659239842726084
      y  y_hat
0     0      1
1     0      1
2     0      0
3     1      1
4     0      1
...  ..    ...
3047  0      0
3048  0      0
3049  0      0
3050  0      1
3051  0      0
```

6.3 Neural Networks

The predictor information is combined by neural networks in a fairly flexible fashion that captures the complex correlations between these factors and between them and the result variable.

Forward propagation is the method used to compute and store intermediate variables for a neural network in the right order from the input layer to the output layer. Now, let's step-by-step walk through how a neural network with a single hidden layer works. Through weights (W) and a bias (b), each neuron is linked to all the neurons in the previous layer. Mathematically it can be calculated as below:

a1 = w1*x1 + w2*x2 + b1, where w is weight and b is bias

The procedure of back propagation is used to calculate the gradient of neural network parameters. The method essentially traverses the network from the output to the input layer in reverse order using the chain rule from calculus.

The error function's gradient with regard to the weights and biases is calculated,

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta}$$

The mean squared error serves as the error function in traditional backpropagation.

$$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

With two hidden layers (16, 8) and an output layer with a sigmoid activation function, we created neural networks using Keras. We used relu for the hidden layer's activation function, the Adam optimizer for loss, and binary cross entropy loss for loss. We ran the model for 100 epochs and observed that the training accuracy and validation accuracy were almost at constant values.

After training the model, we did cutoff vs sensitivity/accuracy analysis:

	cutoff	sensitivity	accuracy
0	0.10	0.991872	0.884338
1	0.20	0.978712	0.916462
2	0.40	0.954070	0.937556
3	0.60	0.918849	0.938524
4	0.80	0.849697	0.915882
5	0.95	0.729712	0.863501

For testing, we decided to keep the cutoff as 0.4 and we got the following results:

```
y_pred= (model.predict(stdtestx)>0.4).astype(int)
y_pred=np.squeeze(y_pred,1)
cm=confusion_matrix(test_y, y_pred)
print(cm)
print('sensitivity:', ((test_y & y_pred).sum()/test_y.sum()), 'accuracy:',
  (test_y== y_pred).sum()/test_y.shape[0])
```

```
96/96 [==============================] - 0s 954us/step
[[2298  248]
 [ 141  365]]
sensitivity: 0.7213438735177866 accuracy: 0.8725425950196593
```

The validation accuracy is lower than the training accuracy indicating that the model is overfitting on training data. To solve this problem , we tried to implement regularization techniques in Neural Network by adding a L2 norm and dropout layers. Although it did not affect the accuracy scores, it did lower the accuracy of model on testing data which is undesirable.

7. Results

Since the goal of our model is to determine the likelihood that a customer would purchase a product, we must trade off poor specificity for higher accuracy and sensitivity. Classifying a product's likelihood of purchase is crucial in order to decide whether to invest in targeted marketing initiatives, among other things. Reaching them through marketing may result in more income, even if some consumers who are not going to buy are recognized as potential customers. However, we risk losing potential customer if we incorrectly categorize the customers who are about to make a transaction.

Neural Networks provide the highest accuracy and respectable sensitivity according to the metric comparison displayed above. Although Nave Bayes provides respectable accuracy and sensitivity, it takes a very long time to run.

In order to solve the problem of finding potential customers, real-world data was used in this research. In comparison to the class of customers we were interested in, the data contained more instances of non-subscribing customers.