IE7374 FINAL PROJECT REPORT

Online Shopper's Purchasing Intention
[Group 10]

Sai Vinay Teja Jakku
jakku.s@northeastern.edu

Akarsh Singh
singh.akar@northeastern.edu

Janvi Shetty
shetty.j@northeastern.edu

Anurag Palanki
palanki.a@northeastern.edu

Percentage of Effort Contributed by Student 1: 25%
Percentage of Effort Contributed by Student 2: 25%
Percentage of Effort Contributed by Student 3: 25%
Percentage of Effort Contributed by Student 4: 25%

Signature of Student 1: *Sai Vinay Teja Jakku*
Signature of Student 2: *Akarsh Singh*
Signature of Student 3: *Janvi Shetty*
Signature of Student 4: *Anurag Palanki*

Submission Date: 08/08/22

Table of Content:

Problem Setting & Definition:

Over the past few years, business owners and retailers, big or small, have started taking their shops online to increase their reach, promote their brand and increase sales that cover a wide range of customers. For these businesses, it is essential to identify the customer base and give them enough rewards and incentives to encourage their online purchase. The primary objective of this project is to identify patterns of the customers that browse a website and make a successful purchase. We intend to achieve this by using various Machine learning techniques and selecting the best classification algorithms that successfully classify the revenue promoting customers from other users using the Online Shoppers' Purchasing Intention dataset.

Data Sources:

UCI Repository link:
https://archivebeta.ics.uci.edu/ml/datasets/online+shoppers+purchasing+intention+dataset

Data Description:

Of the 12,330 sessions in the dataset, 84.5% (10,422) were negative class samples that did not end with shopping, and the rest (1908) were positive class samples ending with shopping[1]. The dataset was created so that each session would belong to a different user in a year to avoid any tendency to a specific campaign, special day, user profile, or period. It contains ten numerical and seven categorical columns, with the 8th and target column being the target column.

Name of Attribute	Type and Range	Description
Administrative	Numerical (0 – 27)	Number of Administrative pages visited in the session
Administrative Duration	Numerical (0 – 3398 sec)	Total time spent on these types of pages
Informational	Numerical (0 – 24)	Number of Informational pages visited in the session
Informational Duration	Numerical (0 – 25549 sec)	Total time spent on these types of pages
Product Related	Numerical (0 – 705)	Number of Product Related pages visited in the session
Product Related Duration	Numerical (0 – 63973 sec)	Total time spent on these types of pages
Bounce Rates	Numerical (0 – 0.2%)	Percentage of visitors entered and left without triggering any other requests during that session

Exit Rates	Numerical (0 – 361)	The percentage that was the last in the session
Page Values	Numerical (0 – 0.2%)	The average value for a web page visited before completing an e-commerce transaction
Special Day	Numerical (0 – 1)	The closeness of the site visit time to a specific special day
Month	Categorical (12)	Month of the year
Operating System	Categorical (8)	Type of OS
Browser	Categorical (13)	Type of Browser
Region	Categorical (9)	Type of geographic region
Traffic Type	Categorical (20)	Type of traffic, a medium that links a session
Visitor Type	Categorical (3)	New visitor (1) returning visitor (2) other (3)
Weekend	Categorical (2)	Weekend (1) weekday (0)
Revenue (Target)	Categorical (2)	Revenue generated (1) else (0)

Splitting The Dataset:

The first step before we start training the model is to split the data into train and test data sets. We train the model using the training dataset and evaluate the fit model using the test dataset. We also have split the data into folds for k-fold cross-validation for one of the machine learning models. Splitting the dataset into train, validation and test data helps in removing bias and in hyper-tuning the parameters.

Exploratory Data Analysis and Visualization:

To get a complete overview of the dataset we were dealing with, we created a function that described the dimensions. The description includes the categorical attributes, numerical attributes, number of null values in all the columns and many more.

```
describe(df)
```

```
Name of all columns in the dataframe:
['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRelated', 'ProductRelated_Dura
tion', 'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType',
'VisitorType', 'Weekend', 'Revenue']

Number of columns in the dataframe:
18

Name of all numerical columns in the dataframe:
['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRelated', 'ProductRelated_Dura
tion', 'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'OperatingSystems', 'Browser', 'Region', 'TrafficType']

Number of numerical columns in the dataframe:
14

Name of all categorical columns in the dataframe:
['Month', 'VisitorType', 'Weekend', 'Revenue']

Number of categorical columns in the dataframe:
4

----------------------------------------------------------------------------------------------------

Number of Null Values in Each Column:

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


Number of Unique Values in Each Column:

Administrative              27
Administrative_Duration    3335
Informational                17
Informational_Duration     1258
ProductRelated              311
ProductRelated_Duration    9551
BounceRates                1872
ExitRates                  4777
PageValues                 2704
SpecialDay                    6
Month                        10
OperatingSystems              8
Browser                      13
Region                        9
TrafficType                  20
VisitorType                   3
Weekend                       2
Revenue                       2
dtype: int64
```

Basic Statistics and Measures for Numerical Columns:

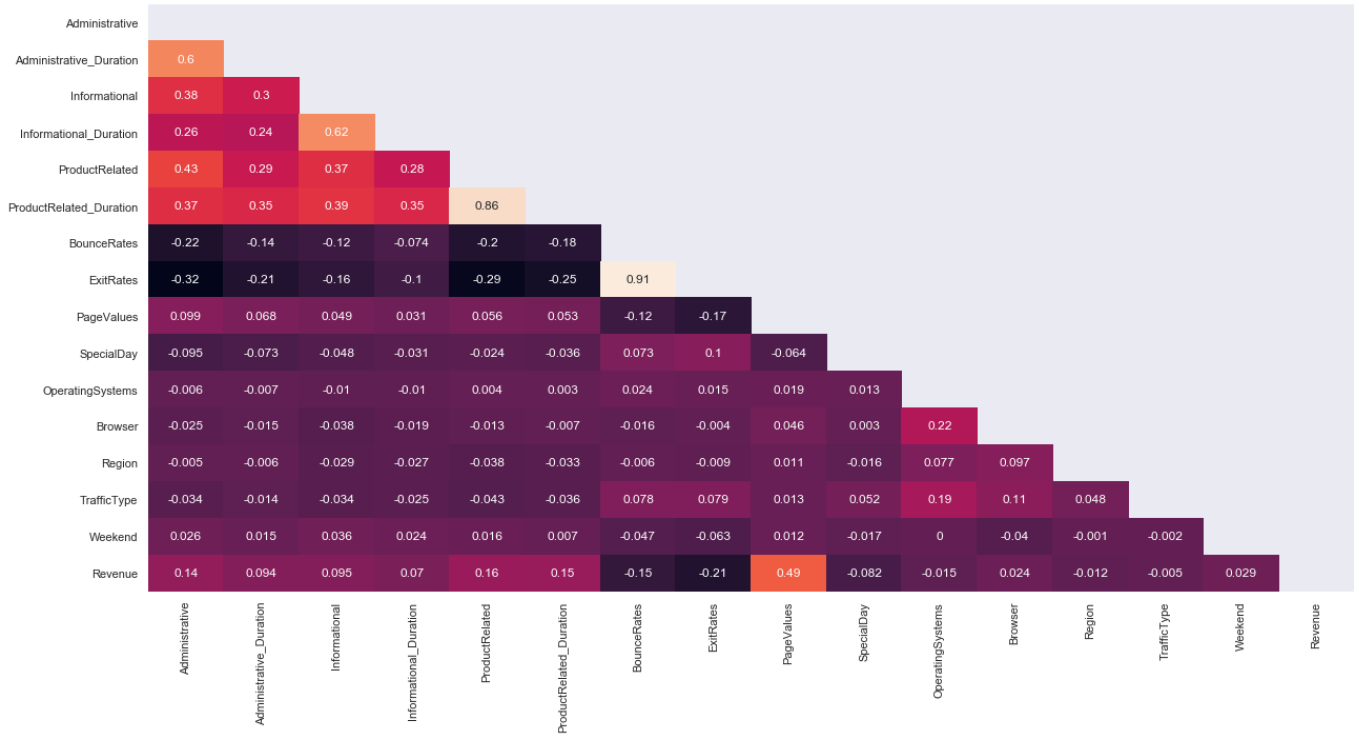
	count	mean	std	min	25%	\
Administrative	12330.0	2.315166	3.321784	0.0	0.000000	
Administrative_Duration	12330.0	80.818611	176.779107	0.0	0.000000	
Informational	12330.0	0.503569	1.270156	0.0	0.000000	
Informational_Duration	12330.0	34.472398	140.749294	0.0	0.000000	
ProductRelated	12330.0	31.731468	44.475503	0.0	7.000000	
ProductRelated_Duration	12330.0	1194.746220	1913.669288	0.0	184.137500	
BounceRates	12330.0	0.022191	0.048488	0.0	0.000000	
ExitRates	12330.0	0.043073	0.048597	0.0	0.014286	
PageValues	12330.0	5.889258	18.568437	0.0	0.000000	
SpecialDay	12330.0	0.061427	0.198917	0.0	0.000000	
OperatingSystems	12330.0	2.124006	0.911325	1.0	2.000000	
Browser	12330.0	2.357097	1.717277	1.0	2.000000	
Region	12330.0	3.147364	2.401591	1.0	1.000000	
TrafficType	12330.0	4.069586	4.025169	1.0	2.000000	

	50%	75%	max
Administrative	1.000000	4.000000	27.000000
Administrative_Duration	7.500000	93.256250	3398.750000
Informational	0.000000	0.000000	24.000000
Informational_Duration	0.000000	0.000000	2549.375000
ProductRelated	18.000000	38.000000	705.000000
ProductRelated_Duration	598.936905	1464.157213	63973.522230
BounceRates	0.003112	0.016813	0.200000
ExitRates	0.025156	0.050000	0.200000
PageValues	0.000000	0.000000	361.763742
SpecialDay	0.000000	0.000000	1.000000
OperatingSystems	2.000000	3.000000	8.000000
Browser	2.000000	2.000000	13.000000
Region	3.000000	4.000000	9.000000
TrafficType	2.000000	4.000000	20.000000

Other Relevant Metadata Regarding the Dataframe:

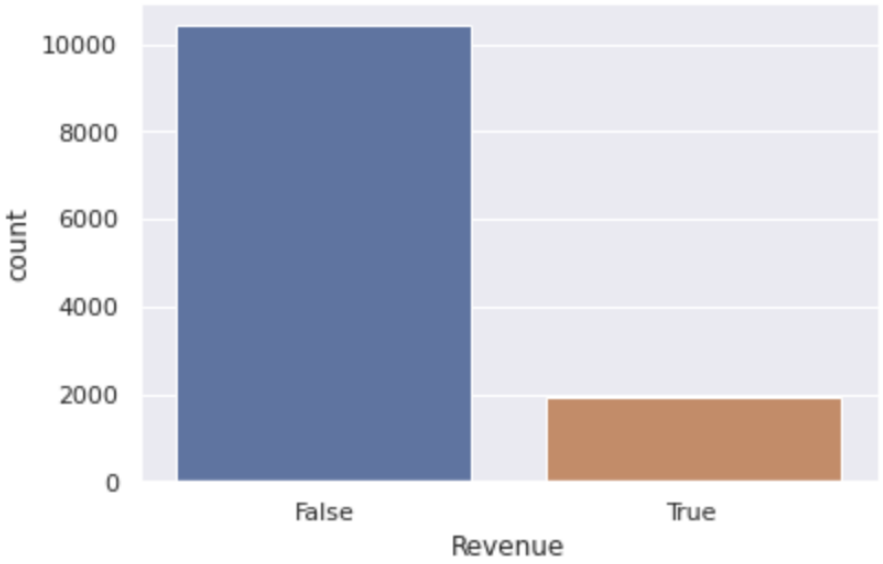
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 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
None
```

All the attributes of the dataset contained non-null values. Hence, null value processing was not required. We then shifted our focus towards finding the correlation of each attribute with every other attribute in the table to remove recursive information.



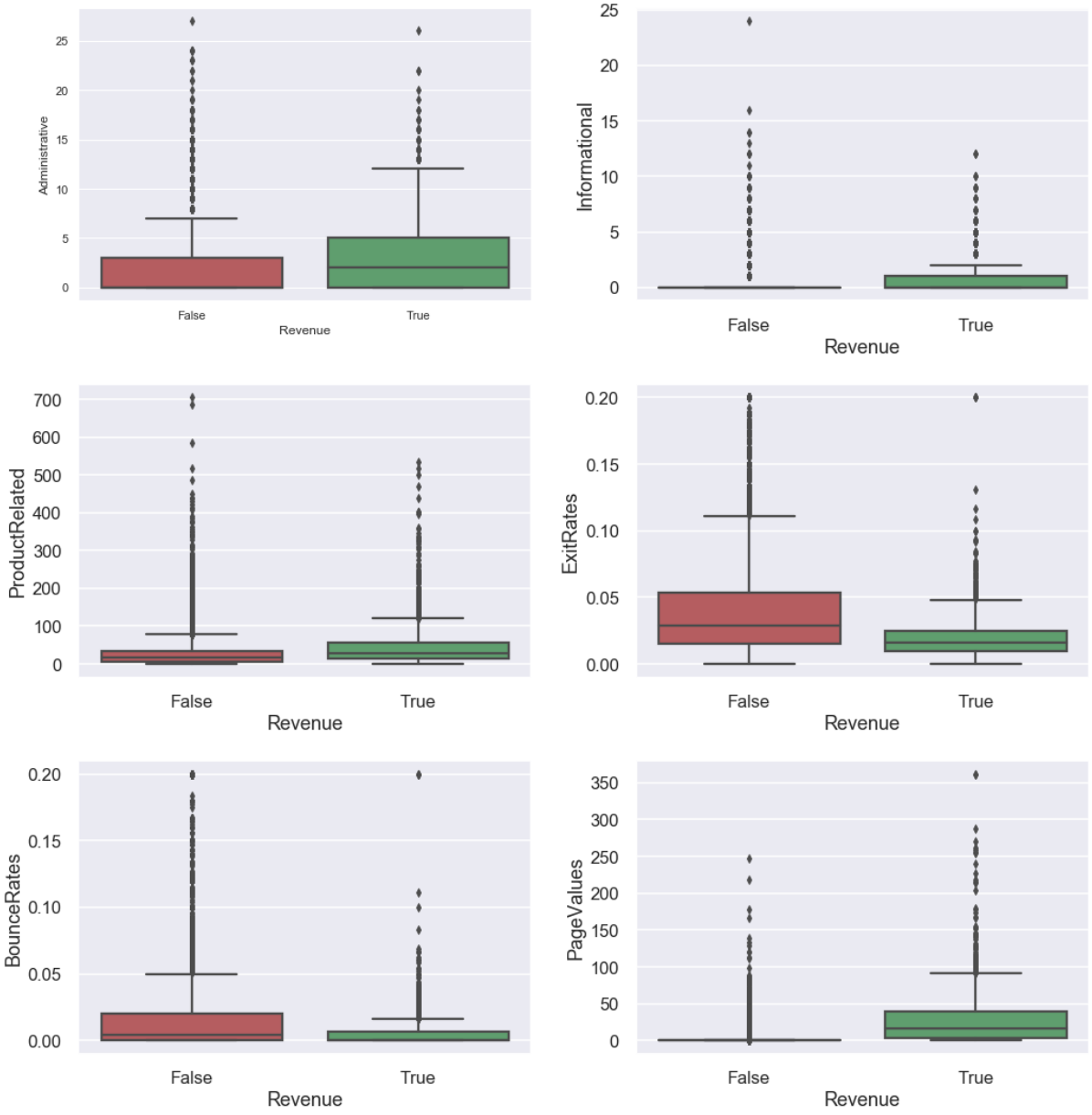
We see a high correlation between the ‘BounceRate’ and the ‘ExitRates’ (0.91) and between the ‘ProductRelated_Duration’ and the ‘ProductRelated’ (0.86). These columns were dealt with while Feature selection.

We plotted a bar plot to find the count values of the two classes in our dataset.



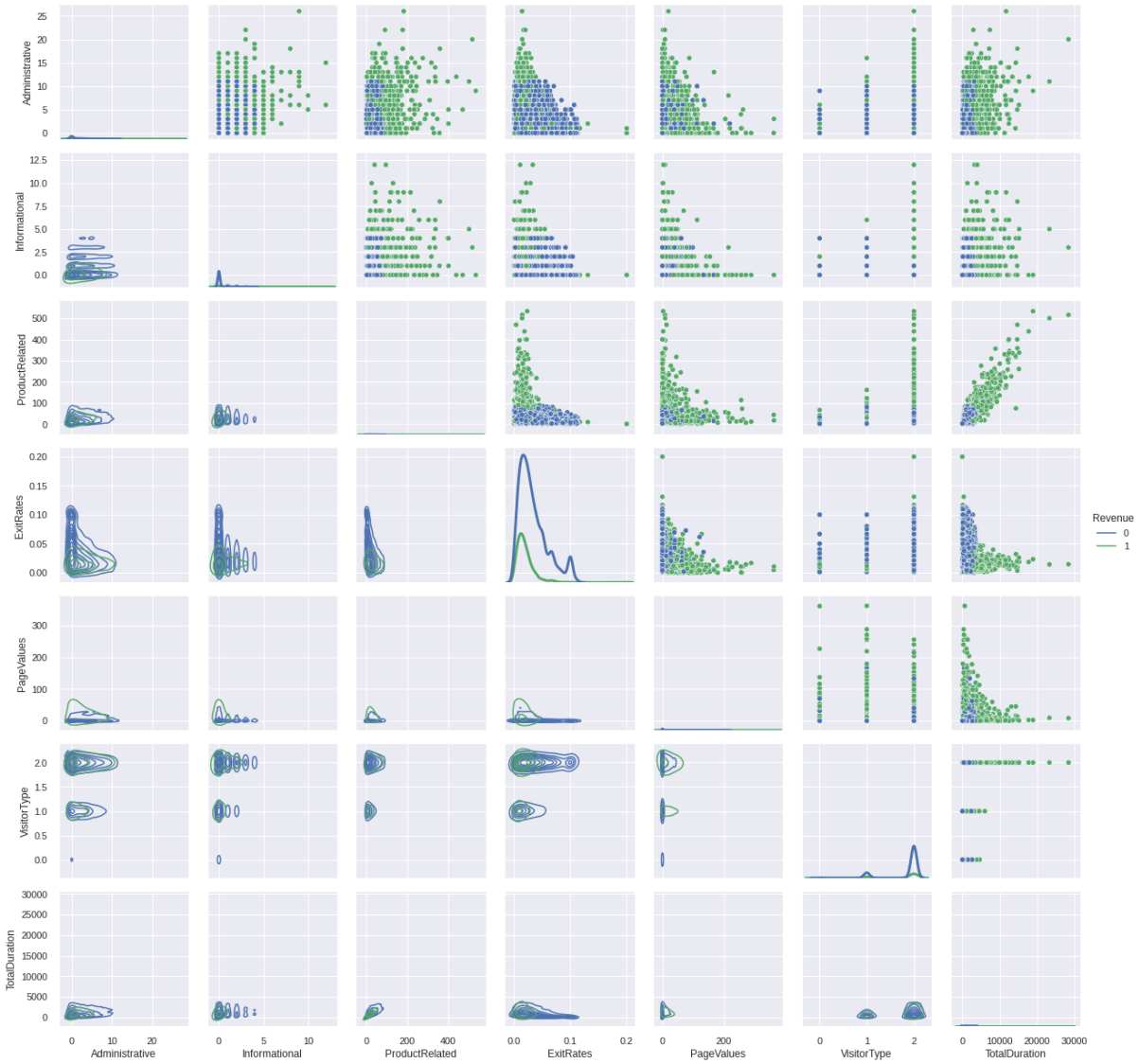
The number of false cases heavily outweighs the number of True classes. From this graph, we realised that we had to work with the skewness in data by either upsampling or down-sampling the data while passing the training data sets into the model.

To evaluate the distribution of values of the columns with relatively high correlation values to the ‘Revenue’ column, a bar plot was used.



The data is highly skewed concerning revenue with many outliers. The skewness of the columns is dealt with while the skew handling process and the outlier and dealt with during Data preprocessing and cleaning.

In the next part, we checked for the linear separability of the two classes for the subset of the total number of attributes in our dataset. We used a grid plot for this case where the lower diagonal shows the kernel density estimate of the distribution concerning the pair of attributes, the upper diagonal shows the scatter plot of the same pair of attributes and the diagonal shows the distribution of values of the attribute across the dataset using kernel density estimate plot. The ‘TotalDuration’ here denotes the summation of the ‘Administrative_Duration’, the ‘Informational_Duration, and the ‘ProductRelated_Duration’.



From the data, we can see that the concentration of the 2 classes and mostly overlapping in all the cases. To the best of our knowledge, we see no separability of these classes when represented in 2-dimensional space.

Feature Selection:

On the basic descriptive statistical analysis of the dataset with respect to the target class (Revenue), it is seen that a few problematic instances have to be addressed that are intrinsic to the dataset.
First, there is a major class imbalance, approximately 1:5 to the class of interest. This calls for data transformation and stringent feature selection to create a generalised model capable of correctly identifying the minority class. (i.e. the class of interest)
Second, almost all the non-categorical variables are significantly skewed to the right and possess significant outliers. Unless taken care of, the model will have significant bias, would not be able to generalise well, and would be susceptible to missing out on tricky instances of minority class data points.
Third, on performing correlation analysis all features present are either weakly correlated or uncorrelated with respect to the target class, except one. If features are not created or data processing is not performed to somehow increase correlation bias would increase with respect to model training and performance will be compromised.
A good thing about the dataset is that there are not any null values hence, dealing with null values prior to processing is altogether eliminated.

The following steps were taken after multiple iterations of trial, and testing, and based on positive statistical impact with respect to change

1. All numerical variables with a Pearson correlation coefficient in the range [-0.1,0.1] with respect to target variable (y) were dropped as their impact on prediction would be insignificant.

2. Attributes like administrative and administrative duration were highly correlated, this will lead to a multi-collinearity issue that is undesirable for model training, hence one variable from 3 pairs of similar attributes was dropped.

3. We created a new feature by performing a weighted sum of all the duration variables and called it ‘total_duration’. These 3 variables were then dropped as addressed in the previous point.

4. The only categorical variable left, based on analysis and visualisations with respect to the output class (y) could be classified as ordinal categorical. The 3 classes within were converted to values (2, 1, and 0) as per priority.

5. This process was followed by outlier detection and removal. We visualised each variable with respect to the target class as a stacked histogram with a KDE plot before removing unwanted data points. We noticed that almost all variables are right-skewed and mostly because of the majority class. Due to the nature of our imbalanced dataset, we only removed outliers from target label 0 to reduce skewness and improve the imbalance ratio. The method used for removing outliers was that of quartiles. The steps are shown below.
    - Find q25th, q50th, and q75th quartile along with interquartile range (IQR, i.e. q75th – q25th quartile)

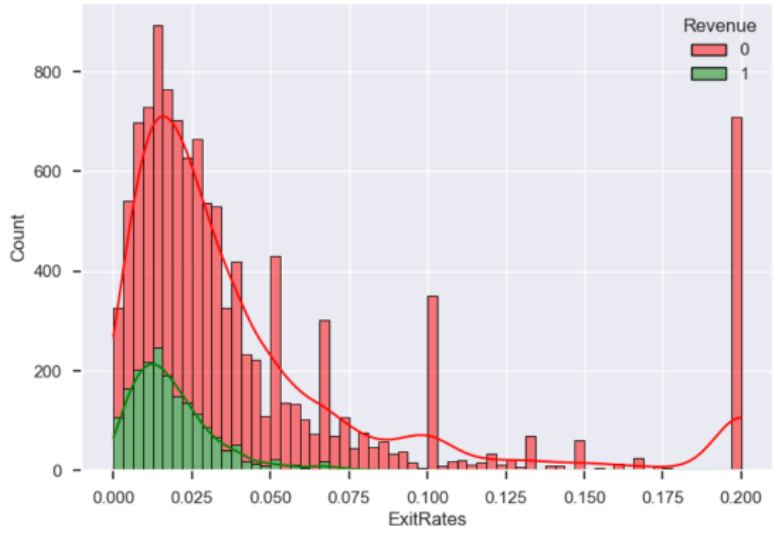
- A constant k (1.5 -2.5) is chosen as a multiplicative factor to be multiplied by IQR to determine scrutiny of outliers. This is subjective to the situation and data. Let's call k*IQR as s

- We then classify values to the left of q25th – s and that to the right of q75th + s as outliers and remove them

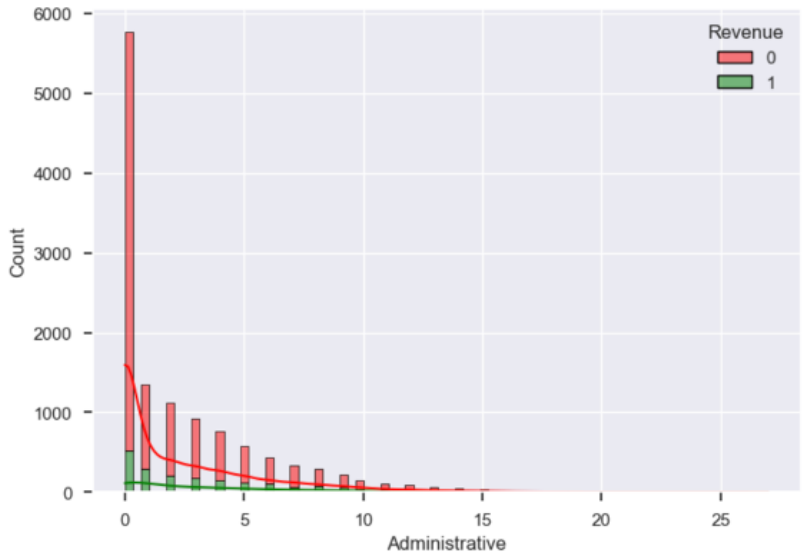
- Since our dataset is skewed to the right, there were no outliers whatsoever beyond q25th – s, when then removed all outliers belonging to class 0 of the target label (y).

A couple of examples are shown below for outlier detection.

```
q75:  0.05
q25:  0.014285714
Inter Quartile Range:  0.035714286000000005
Outliers lie before -0.05714285800000001 and beyond 0.11428571480000001
Number of Rows with Left Extreme Outliers: 0
Number of Rows with Right Extreme Outliers: 1040
```




```
q75:  4.0
q25:  0.0
Inter Quartile Range:  4.0
Outliers lie before -8.0 and beyond 11.2
Number of Rows with Left Extreme Outliers: 0
Number of Rows with Right Extreme Outliers: 299
```



6. Lastly the data frame was normalised to a standard scale before feeding it into a model.

Post outlier treatment 17.28% of the dataset size was reduced where 2131 data points from majority class 0 were dropped. The final correlation map looked as follows (Revenue being the target class)

	Administrative	Informational	ProductRelated	ExitRates	PageValues	VisitorType	TotalDuration	Revenue
Administrative								
Informational	0.34							
ProductRelated	0.37	0.32						
ExitRates	-0.31	-0.13	-0.28					
PageValues	0.09	0.06	0.08	-0.20				
VisitorType	-0.05	0.05	0.13	0.22	-0.12			
TotalDuration	0.40	0.38	0.86	-0.25	0.10	0.11		
Revenue	0.21	0.20	0.32	-0.23	0.50	-0.08	0.35	

Principal Component Analysis:

PCA is an exploratory data analysis tool to reduce dimensions of the data to its important features. We implemented PCA to explain the variance observed between the features in the data. We did this by writing a PCA function which takes the data, number of PCA components as input and returns reduced features data and the eigenvalues (variance captured) . The steps include calculating mean, and standardising the data. We then calculated the eigenvectors and eigenvalues and sorted them in descending order. The intuition is that the eigenvectors explain directions of the data and the eigenvalues represent the magnitude of importance for these directions and we try to capture those important features which explain the spread of the data. We eliminate the less important ones and achieve dimensionality reduction.

We observed that approx 93% of the variance is captured by 3 features and 99.99% of variance is captured by 5 features. Since the variance captured is the same before and after PCA, we decided to move ahead without reducing the dimensions.

```
def PCA(X , num_components):

    X_meaned = X - np.mean(X , axis = 0)
    cov_mat = np.cov(X_meaned , rowvar = False)
    eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)
    sorted_index = np.argsort(eigen_values)[::-1]
    sorted_eigenvalue = eigen_values[sorted_index][0:num_components]
    sorted_eigenvectors = eigen_vectors[:,sorted_index]
    eigenvector_subset = sorted_eigenvectors[:,0:num_components]
    X_reduced = np.dot(eigenvector_subset.transpose() , X_meaned.transpose() ).transpose()
    return X_reduced, sorted_eigenvalue
```

```
X_PCA, lambdas = PCA(df.values[:, 0:-1],5)
```

```
for i in lambdas:
    print(i/lambdas.sum())
```

```
0.5443850226676203
0.28787163201095883
0.09905448144544234
0.04039349848973198
0.028295365386246506
```

Upscaling:

Since the target class is a minority class, we have decided to upscale the target class using the Synthetic Minority Oversampling Technique also known as SMOTE to balance the dataset. It is done before fitting the data to the model and does not give any additional information to the model.

‘SMOTE first selects a minority class instance ‘a’ at random and finds its k nearest minority class neighbours. The synthetic instance is then created by choosing one of the k nearest neighbours 'b’ at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a CONVEX combination of the two chosen instances a and b’[2]
This data augmentation technique is effective because it creates samples that are relatively close in feature space to existing examples from the minority class.

Model Implementation:

Following are the different Machine Learning Models we Implemented For Classification:

1. Logistic Regression
2. Gaussian Naive Bayes
3. Neural Network

1. Logistic Regression

It's a form of statistical software that analyses the association between a dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation.

$$h_\Theta(X) = \frac{1}{1 + e^{-\Theta X}}$$

This analysis can help anticipate the likelihood of an event or a choice occurring. The output obtained in the case of logistic regression is always between (0 and 1), which is suitable for a binary classification task. The higher the value, the higher the probability that the current sample is classified as class= 1, and vice versa. It takes both continuous and discrete variables as input. The data fed into the model is upscaled on the minority class.

Cross Validation:

Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations[2]. It is a robust measure to prevent overfitting[2]. The complete dataset is split into parts. In standard K-fold cross-validation, we need to partition the data into k folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining holdout fold as the test set. In our case, we chose the k to be 5.

Algorithm:

1. The dataset values are split into training and testing sets. The training set is up-scaled using the smote function to take into account the biassed number of true to false values in the target variable.
2. The training data is split into 5-subsets on which 5-fold validation is performed. An array is created to store the gradient calculated for each cross-validation.
3. In every fold we initialize the gradient term to zero and pass it to a function that calculates the optimal gradient term taking into account the maximum number of iterations.
4. A threshold is set while calculating the change in the gradient taking into account the set tolerance value by the user to prevent overfitting.
5. The set value is added into the array initially defined for storing the optimal gradient in each fold.
6. The gradient values are then averaged to the number of folds taken for cross-validation, and this gradient term is used for the evaluation of the validation data.

Logistic Regression Results:

We tried tuning the parameters such as learning rate and tolerance here are a few of them

Tolerance	learning rate	accuracy	F1	recall	precision
0.00001	0.001	0.9023529412	0.7060212515	0.6749435666	0.7400990099
0.001	0.01	0.9050980392	0.7352297593	0.7584650113	0.7133757962
0.01	0.1	0.8968627451	0.7362086259	0.8284424379	0.6624548736
0.0001	0.001	0.9023529412	0.7060212515	0.6749435666	0.7400990099
0.0001	0.0001	0.8956862745	0.6453333333	0.546275395	0.7882736156

The balanced results are given with a learning rate of 0.1 and tolerance of 0.01 concerning the target class i.e Revenue.

The best results were given with a learning rate of 0.1 and a tolerance of 0.01. The test accuracy obtained was 0.89, with precision and recall as 0.66 and 0.82. This is a much better balance when compared to our baseline logistic regression model (threshold 0.5) which had a precision and recall of 0.78 and 0.47 respectively whereas our model gave the best results with a threshold value of 0.9.

2. Gaussian Naive Bayes

Gaussian Naive Bayes assumes that each class follows a Gaussian distribution. As our dataset contained two classes (True and False), the Dataset did not have to go through any manipulation to be fed into the code. Priors were simple. We parsed the target value of the training set which was upscaled before the calculation to find the average number of true and false values in it.

We already assumed that the distribution of values in the columns is gaussian, so for each dimension, mean and standard deviations are calculated. For any new point, we look into the probability distribution of the dimension values to find the probability of it belonging to the respective distribution. The function used is

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Algorithm:

1. The dataset values are split into training and testing sets. The training set is up-scaled using the smote function to take into account the biassed number of true to false values in the target variable.
2. The training data is split data comprising the two classes, which are to be classified.
3. The priors and distribution for respective classes are calculated using the sub-datasets that were taken in the previous step.
4. Once the probability values are calculated for each class, the calculated values are then used for the prediction of the test set.

Gaussian Naive Bayes Results:

Metrics for result comparison used are the Accuracy, Precision, Recall and the F1 score:

```
Model Accuracy is:  0.1784313725490196
Model Precision is:  0.18041237113402062
Model Recall is:  0.9420289855072463
F1 score is:  0.30282861896838603
```

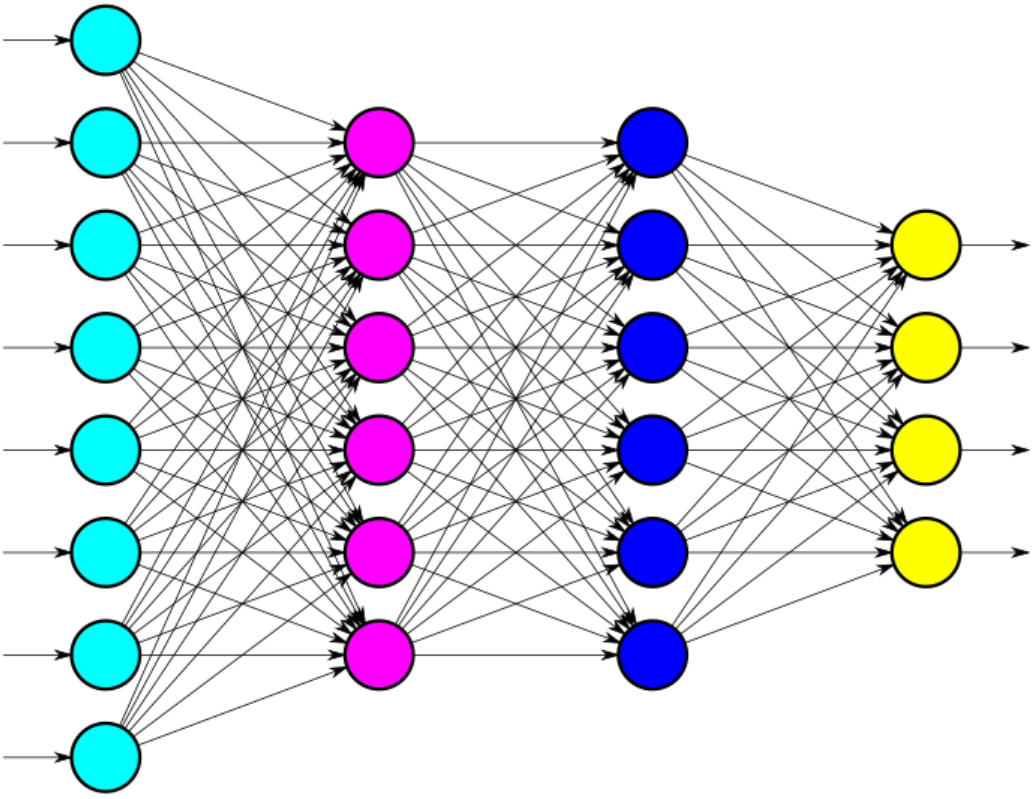
The Gaussian Naive Bayes gave poor performance values compared to our baseline model (which is our Logistic regression model with a 0.5 threshold). It had a test accuracy and precision of 0.17 and 0.18 respectively and a low F1-Score of 0.3 with respect to target class 1. Our baseline model had precision and recall of 0.78 and 0.47 respectively. These stats were expected considering the imbalance in the test data.

```
cm = confusion_matrix(clfGNB.y_test, y_pred)
print(' |-----Act-----')
#print('--0-------1--')
print('P|---0------1--')
print('r|0',cm[0])
print('e|1',cm[1])
```

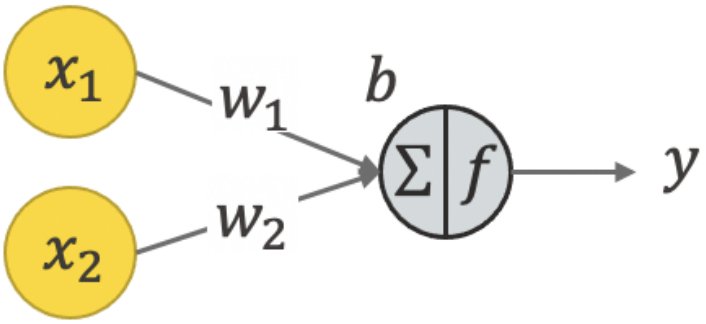
```
 |-----Act-----
P|---0------1--
r|0 [   0 2067]
e|1 [ 28 455]
```

The model recall value is high because of the highly skewed classification. As seen above, the model tends to classify the majority of the observations to class 1 which is our target class, which is not a desirable outcome. Thus, from the results seen here, this model should not be considered for our dataset given the highly skewed distribution.

1. Neural Networks



Here, we have used fully connected neural networks with 1 input layer, 2 hidden layers, and a final output layer when the sigmoid function is applied to give a probability between 0 and 1 for the respective classes. This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications, the units of these networks apply a sigmoid function as an activation function. Other functions include tan(h), relu, etc. A basic structure of a single node is displayed below:

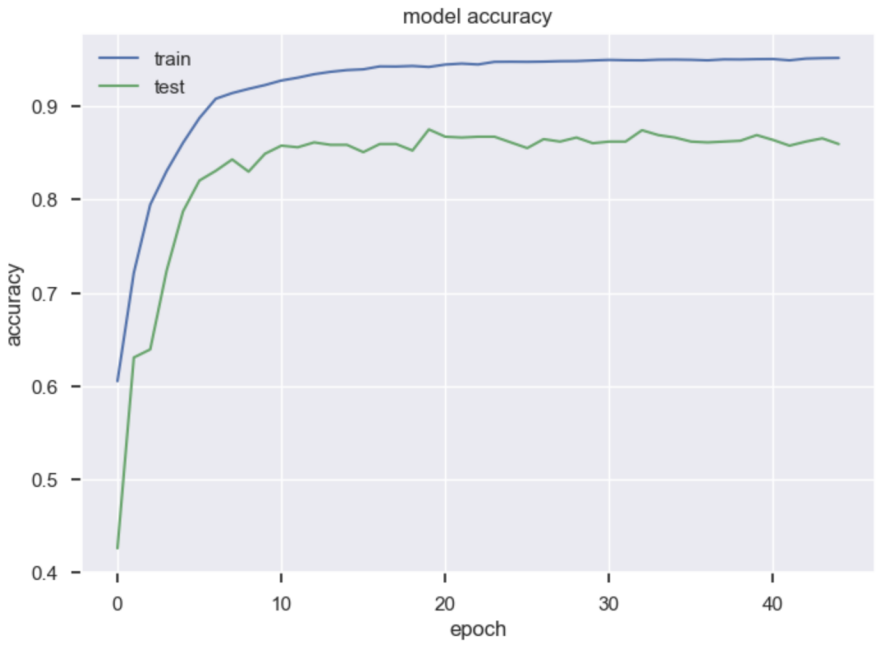


The model summary for the final model used is also shown below

```
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_114 (Dense)            (None, 64)                512
_________________________________________________________________
dense_115 (Dense)            (None, 32)                2080
_________________________________________________________________
dense_116 (Dense)            (None, 16)                528
_________________________________________________________________
dense_117 (Dense)            (None, 8)                 136
_________________________________________________________________
dense_118 (Dense)            (None, 1)                 9
=================================================================
Total params: 3,265
Trainable params: 3,265
Non-trainable params: 0
```

Bias-Variance Trade-off:

We monitored the validation loss in tandem with the training loss and the number of epochs to see how it changes. Initially starting with a higher number of iterations (epochs) we noticed that the validation loss is increasingly fluctuating from a minimum point, this meant that the model was overfitting. We balanced that off by reducing the number of epochs. After that, we noticed that the validation accuracy was increasing which meant that the model would generalise well.



Neural Networks Results:

The neural network was our best-performing model that generalised well. It had a test accuracy of 0.90 but more importantly precision and recall of 0.74 and 0.75 respectively with respect to target class 1. This is a much better balance when compared to our baseline logistic regression model (threshold 0.5) which had a precision and recall of 0.78 and 0.47 respectively. These stats are good considering the sheer class imbalance in the test data. In simpler terms, the model is capable of recognizing every 3/4 true labels correctly. Accuracy is high because many 0-class data points can be identified easily.

Some additional results obtained are shown below in the form of a table

epochs	batch size	accuracy	Precission	Recall	F1 score
50	256	0.8784313725	0.72319202	0.5930470348	0.6516853933
150	512	0.8925490196	0.7292110874	0.6993865031	0.7139874739
150	256	0.9003921569	0.7317554241	0.7586912065	0.7449799197
150	64	0.8941176471	0.7151277014	0.7443762781	0.7294589178

Final Results:

The granularity and nature of the dataset bought their challenges that needed to be handled to generate a generalised usable predictive model. Through the processes, most time was spent on understanding nuances of the features from a statistical standpoint and selecting the best ones based on evidence and standard procedures. Upscaling was also important to elevate our class of interest and SMOTE does a good job in the mix for the same. We included multiple classification algorithms and gauged the pros and cons for each. Fully connected neural networks outperforms other algorithms for our case by having high precision and recall values that are also close to each other. Logistic regression also gave us good results. We also note that the best results were obtained when the threshold value for positive classification was over 0.8. This shows that due to the sheer amount of class imbalance in the test set, it is better off to classify a datapoint as positive until the model is highly certain. As not a part of our course, decision tree based models were not tested for this project.

Citation:

[1] Sakar, C. & Kastro, Yomi. (2018). Online Shoppers Purchasing Intention Dataset, UCI Machine Learning Repository
[2] Page 47, Imbalance Learning: Foundations, Algorithms, and Applications, 2013