ONLINE SHOPPERS PURCHASING INTENTION CLASSIFICATION

Submitted by	Submitted To
Bardia Mouhebat	Prof. Ramin Mohammadi
Charunya Ganganna	
Moheth Muralidharan	
Raghul Sekar	

Abstract

The goal of the project is to predict if online shoppers end up shopping, after each session online based on different session predictors. The problem is classified by two determining factors, Positive- if the shopper ends up shopping and Negative- if they do not.

Introduction

The online shoppers purchasing intention classification dataset contains the information of each session the shopper visits online, based on which a prediction is made whether a person is shopped/ not shopped. We use 3 models to classify the observations. First, we propose a baseline model and then implement logistic regression, naïve bayes and neural networks to classify the results.

Data Description

The online shoppers purchasing intention classification data set contains18 columns and 12330 rows of different users shopped online over a span of a year, which included 17 features variable and 1 target variable “Revenue”. The goal of this project is to predict variable Revenue by using their relevant features. Out of the total 12330 session, around 84.5% are negative class and 15.5% are positive class. Among 17 features for predicting the class, 10 were numerical and 7 categorical variables.

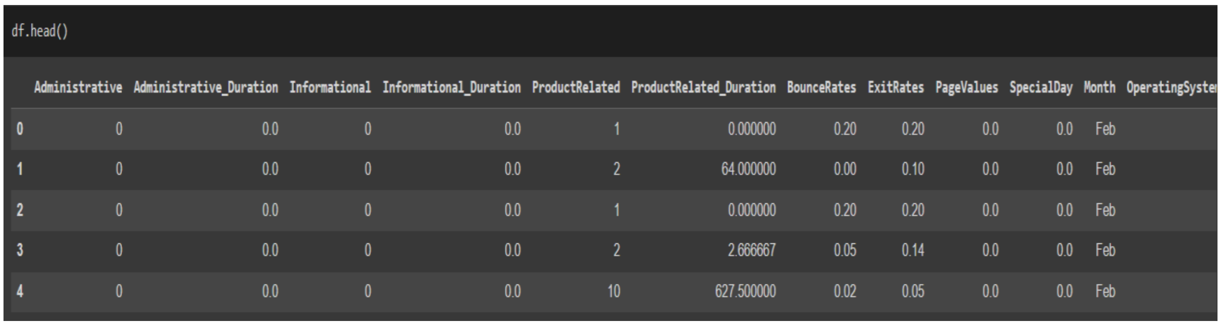
The description of each variable is shown in the table below.

Predictor	***Type***	***Description***
Administrative	Numerical	Number of administrative pages visited during the session
Administrative Duration	Numerical	Total time spent in administrative pages
Informational	Numerical	Number of informational pages visited during the session
Informational Duration	Numerical	Total time spent in informational pages
Product Related	Numerical	Number of product related pages visited during the session
Product Related Duration	Numerical	Total time spent in product related pages
Bounce Rate	Numerical	The percentage of visitors who enter the website through that page and exit without triggering any additional tasks.
Exit Rate	Numerical	The percentage of pageviews on the website that end at that specific page.
Page Value	Numerical	The average value of the page averaged over the value of the target page and/or the completion of an eCommerce
Special Day	Numerical	This value represents the closeness of the browsing dateto special days or holidays
Month	Categorical	Contains the month of the session in string form
Operating Systems	Categorical	An integer value representing the operating system that the user was on when viewing the page.
Browser	Categorical	An integer value representing the browser that the user was using to view the page.

Region	Categorical	An integer value representing which region the user is located in.
Traffic Type	Categorical	An integer value representing what type of traffic the user is categorized into.
Visitor Type	Categorical	A string representing whether a visitor is New Visitor, ReturningVisitor, or Other.
Weekend	Categorical	A Boolean representing whether the session is on a weekend.

Data Exploration and Cleaning

Data cleaning and exploration includes handling raw data and presenting it in a usable, understandable format.
To get an overview of the data and their datatypes, we load the first five rows.

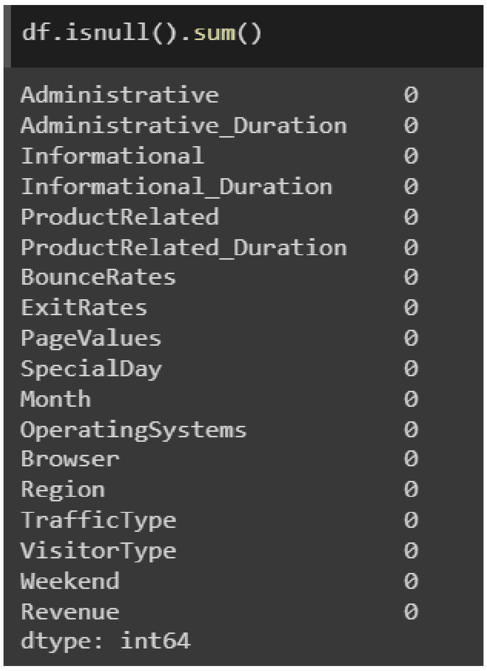


```
df.head()
```

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

Overview of the data

Analyzing null values in each of the columns is one of the necessary steps when we are cleaning the data.



```
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
```

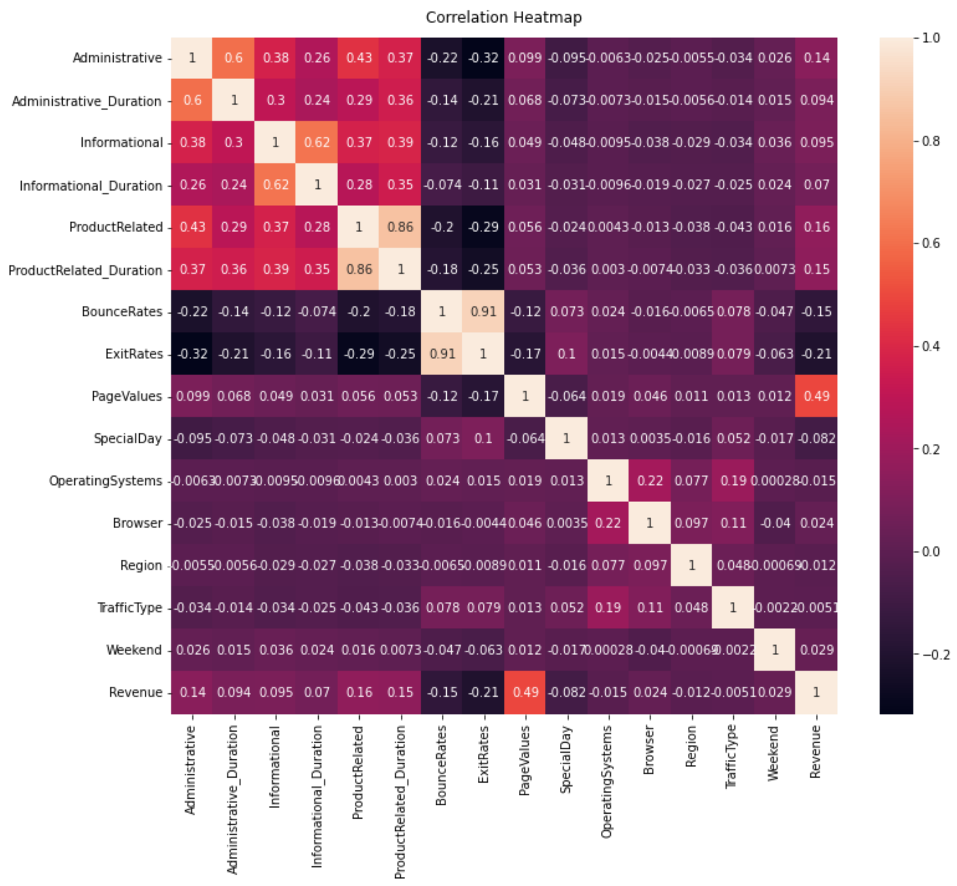
Null Values

As we can see, there are no null values seen. As a result, we don't need to change this dataset in any way.

Data Exploration and Visualization

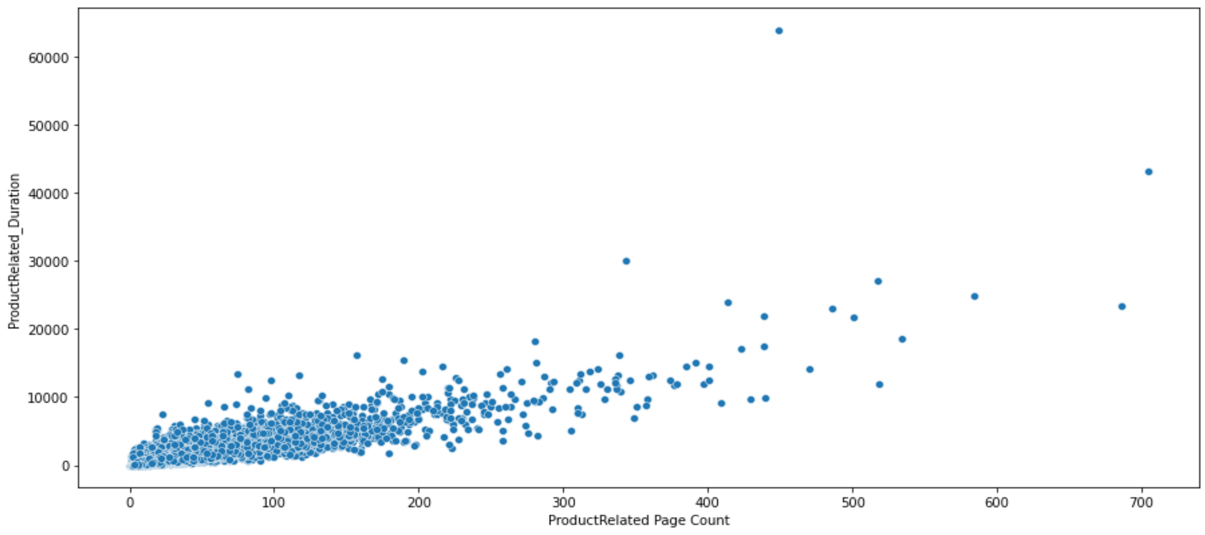
Data exploration and visual representation are important tools for telling a story and making it simple to understand by showing trends.

To check how each variable is correlated with each other, we have plotted a correlation plot.

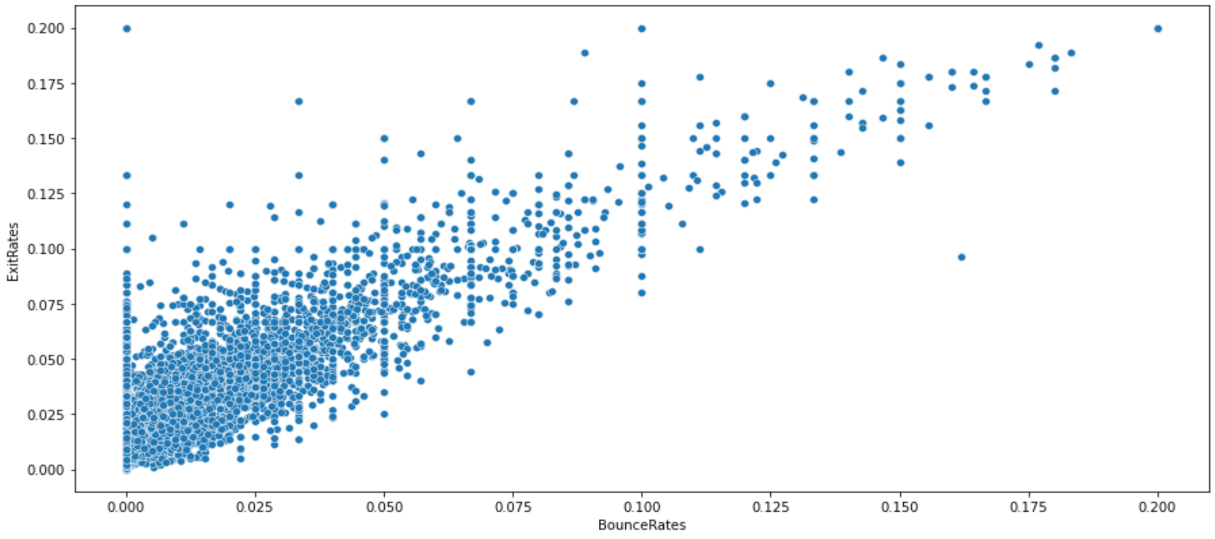


Correlation Plot

As we can see, two sets of variables “BounceRates” - “ExitRates” and “ProductRelated”- “ProductRelated_Duration” have correlation values greater than 0.8. To understand better, we have plotted scatter plot against the highly correlated variables.

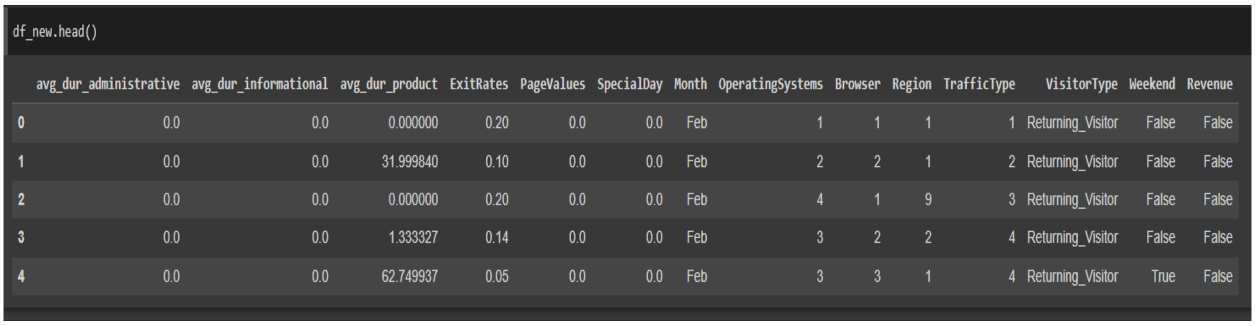


Scatter plot- ProductRelated vs ProductRelated_Duration



Scatter plot- BounceRated vs ExitRates

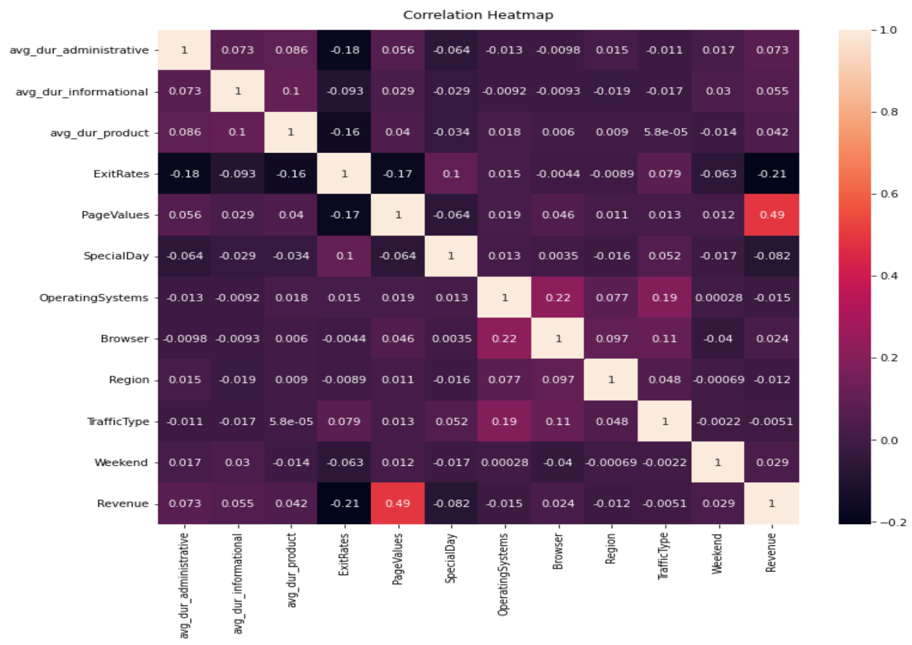
With these highly correlated variables, we can create or remove features from the existing data set. Here we create three new features out of Informational, Administartive and ProductRelated categories by removing the BounceRate feature.



```
df_new.head()
```

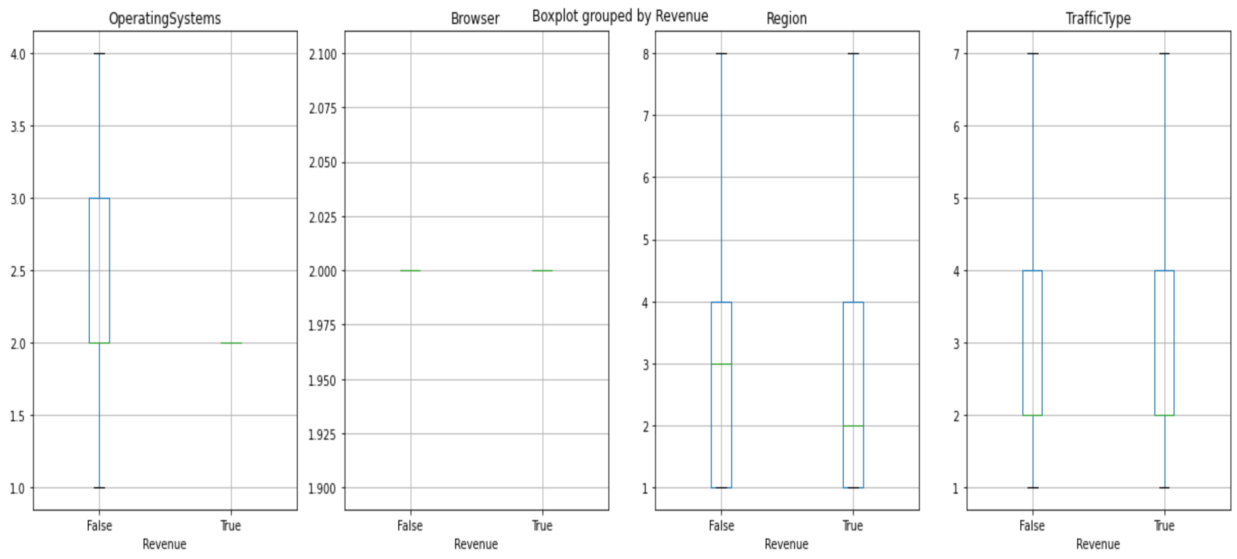
	avg_dur_administrative	avg_dur_informational	avg_dur_product	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0	0.0	0.0	0.000000	0.20	0.0	0.0	Feb	1	1	1	1	Returning_Visitor	False	False
1	0.0	0.0	31.999840	0.10	0.0	0.0	Feb	2	2	1	2	Returning_Visitor	False	False
2	0.0	0.0	0.000000	0.20	0.0	0.0	Feb	4	1	9	3	Returning_Visitor	False	False
3	0.0	0.0	1.333327	0.14	0.0	0.0	Feb	3	2	2	4	Returning_Visitor	False	False
4	0.0	0.0	62.749937	0.05	0.0	0.0	Feb	3	3	1	4	Returning_Visitor	True	False

The data set description after removing the features



Correlation plot after removing the features

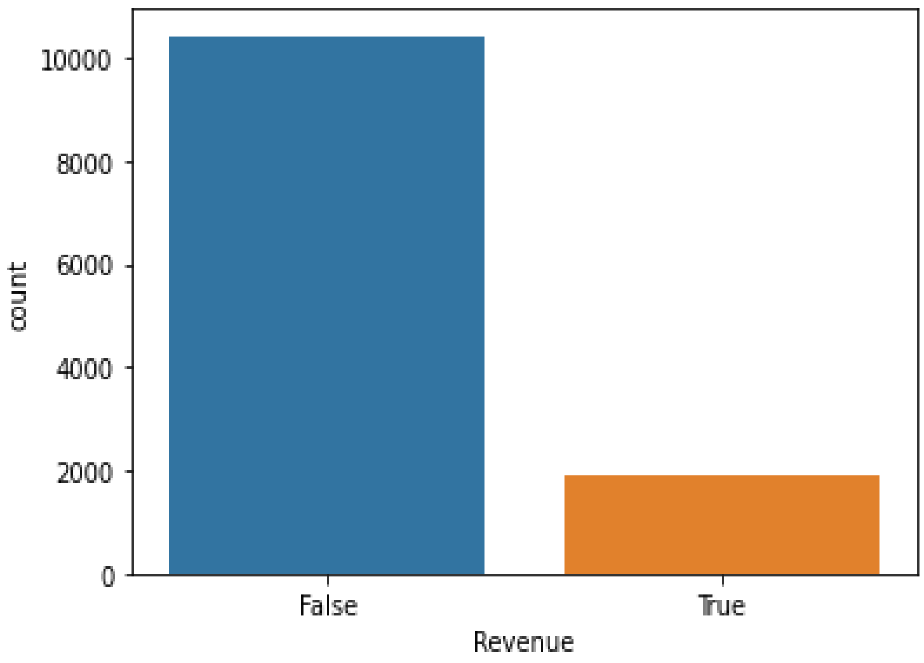
From the above plot, we can observe that OperatingSystem, Browser, Region and TrafficType seem to have very less impact on target variable Revenue.



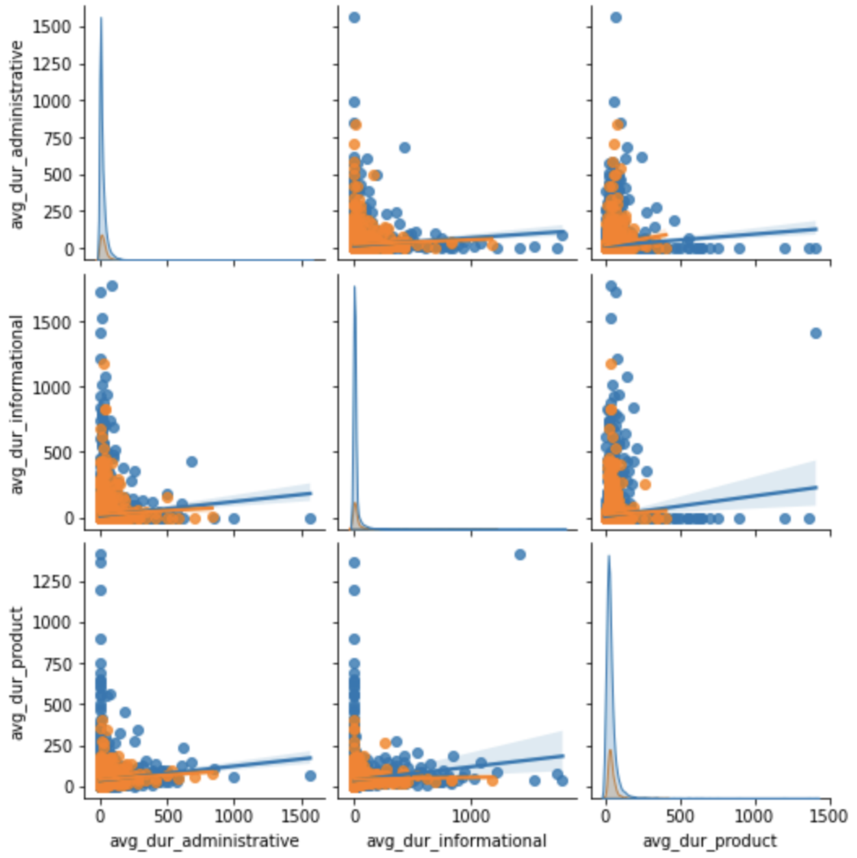
Distribution against Revenue

Below are some of the graphical representations to visualize the dataset-

1. To check how many customers ended up shopping/ not shopping, we plotted a bar plot. We observed that around 85% of the shopper did not shop anything.

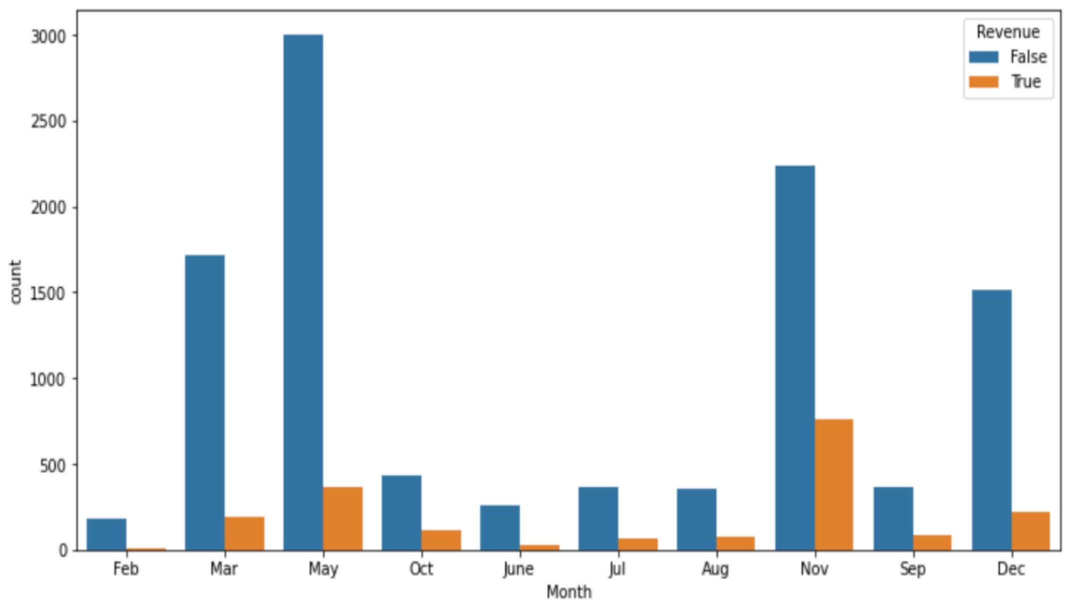


2. We made a pairplot among the newly created variables to see the relationship between them with respect to revenue



.

3. To check month wise distribution of Revenue, we plotted the below grouped bar chart. We can see that most sessions that end up with revenue are in November, this can be due to black Friday being in November.



Methods

1. Logistic Regression:

A logistic regression is a parametric classification model that outputs the probability of a record belonging to the success class. Based on a threshold, we can convert these probabilities into either 0 or 1 thus making the output variable categorical. It is possible to model the outcome variable as a linear function of the predictors

2. Naïve Bayes:

Naive Bayes Performed Gaussian Naïve bayes algorithm since it's a classification model. Basically, it is the calculation of the probabilities for each hypothesis simplified to make their calculation tractable. Given the target value, the attribute values are thought to be conditionally independent and are computed as P(d1|h) * P(d2|H), and so on, rather than trying to compute each attribute value individually, P(d1, d2, d3|h). A list of probabilities can be found in the file of a fitted naive Bayes model.

3. Neural Network:

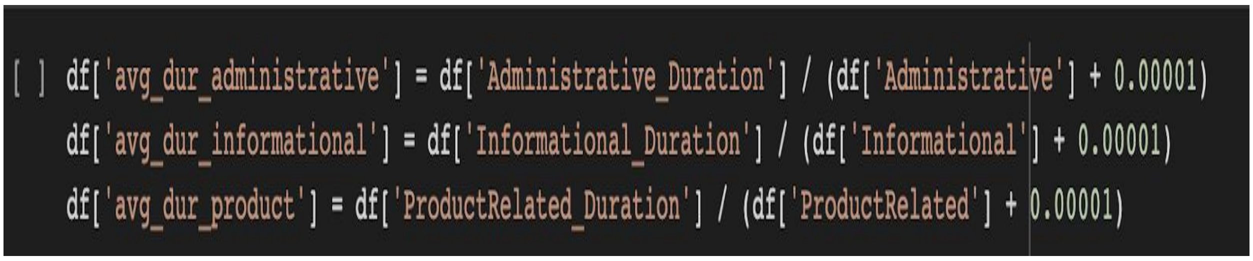
Neural network approach is a computational system that solves problems like a human brain. Neural Networks has an input layer, an output layer and hidden layers between the input and output layers. These layers are like neurons in the human brain. The layers are connected via nodes, and these connections form a network. Neural Networks form the basis of deep learning and have enormous applications in the real world.

Data Preprocessing and Feature Engineering:

We had three categorical variables namely ‘Month’, ‘Visitor Type’ and ‘Weekend’ and we encoded them using the pd.get_dummies() method. Now there are no categorical variables left in the dataset.

The columns ‘BounceRate’ and ‘ExitRate’ had a high correlation coefficient of greater than 0.8 which means they contain the same information. So, we removed the ‘BounceRate’ column from the dataset.

We created 3 new columns by combining ‘Administrative_Duration’ and ‘Administrative’, Informational_Duration and Information,’ProductRelated_duration’ and ‘ProductRelated’ and removed the existing columns to reduce the dimension and redundancy.



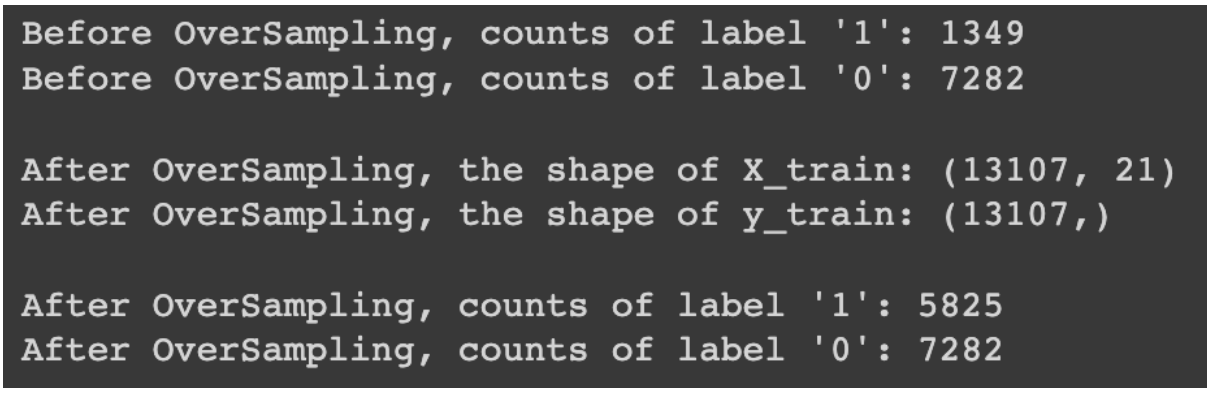
```
[ ] df['avg_dur_administrative'] = df['Administrative_Duration'] / (df['Administrative'] + 0.00001)
    df['avg_dur_informational'] = df['Informational_Duration'] / (df['Informational'] + 0.00001)
    df['avg_dur_product'] = df['ProductRelated_Duration'] / (df['ProductRelated'] + 0.00001)
```

Since OperatingSystem, Browser, Region and TrafficType columns had very little impact on the target, we removed the columns.

Balancing the data

The target class of our dataset, revenue, has a majority class which is False for shopping intention and a minority class which is True for shopping intention. The minority class only accounts for 15% of the observations. This can lead to our model having poor performance. To battle this issue, we decided to use the SMOTE technique.

Synthetic Minority Oversampling Technique creates synthetic data for the minority class so that the target variable is unbalanced and the model trains better. The method does not give any additional information to us. We can perform under sampling for the majority class before creating synthetic data for the minority class. However, for this project we decided to skip this part and only oversample the minority class. After performing SMOTE, we have 13188 training samples from which 7327 were labelled False and 5861 were labelled True. So, 45% of the training data are labelled False. This is way more than 15% that we had in the beginning.



```
Before OverSampling, counts of label '1': 1349
Before OverSampling, counts of label '0': 7282

After OverSampling, the shape of X_train: (13107, 21)
After OverSampling, the shape of y_train: (13107,)

After OverSampling, counts of label '1': 5825
After OverSampling, counts of label '0': 7282
```

Model Selection and Implementation

For this classification task we decided to use 3 models and compare the results. The models that are implemented are Logistic Regression, Naïve Bayes, and Neural Networks. Our initial dataset is divided into 15% observations with class 1 and 85% with class 0. Therefore, a random model should have an accuracy higher than 85% so we can say that our baseline accuracy is 85%. However, we should know that in this task it is more important to classify the target class correctly and for that reason we will use precision, recall and F1-score metrics.

We split the dataset to train, development, and test sets. We will train the models and try to optimize the hyper-parameters based on the validation scores. Finally, we will report the results for the test set.

Logistic Regression

First, we use logistic regression to classify the observations with the binary target variable. The sigmoid function will give us a probability between 0 and 1 and after that we can classify the observations with different thresholds. For this model we try to optimize the learning rate, epsilon, regularization term and threshold so that the F1-score for the validation set is maximized.

We first fit our model with learning rate=0.01, epsilon=0.0001, regularization term=0.001 and threshold=0.5. This will be used as the baseline logistic regression model. The accuracy for this model is 0.86 for the development set and the F1-score is 0.61. For this model we can see that the cost has not reached it minimum point, this can be due to low epsilon for our model. Also, the distribution of the predictions can be seen in figure 2.

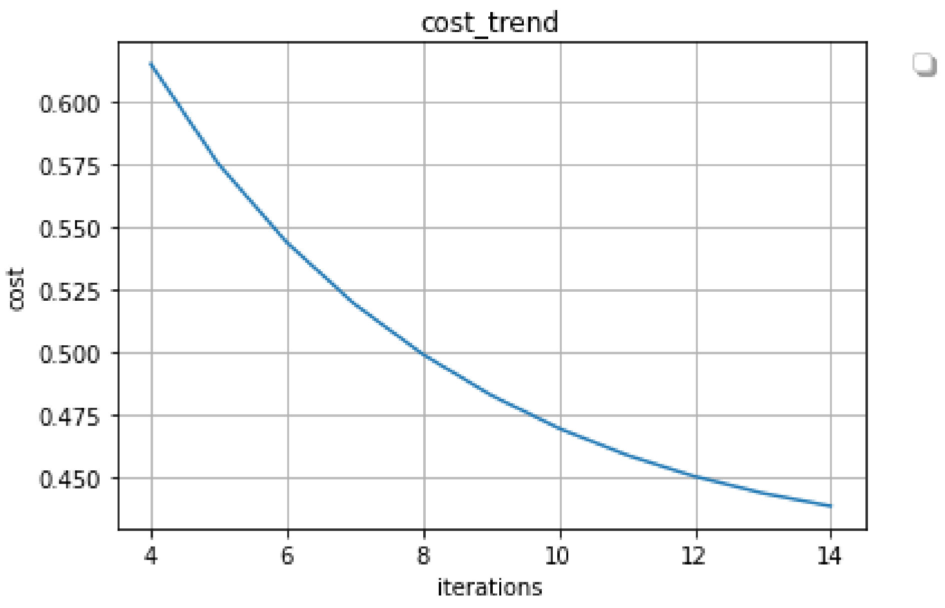


Figure 1 Cost for the baseline logistic regression model

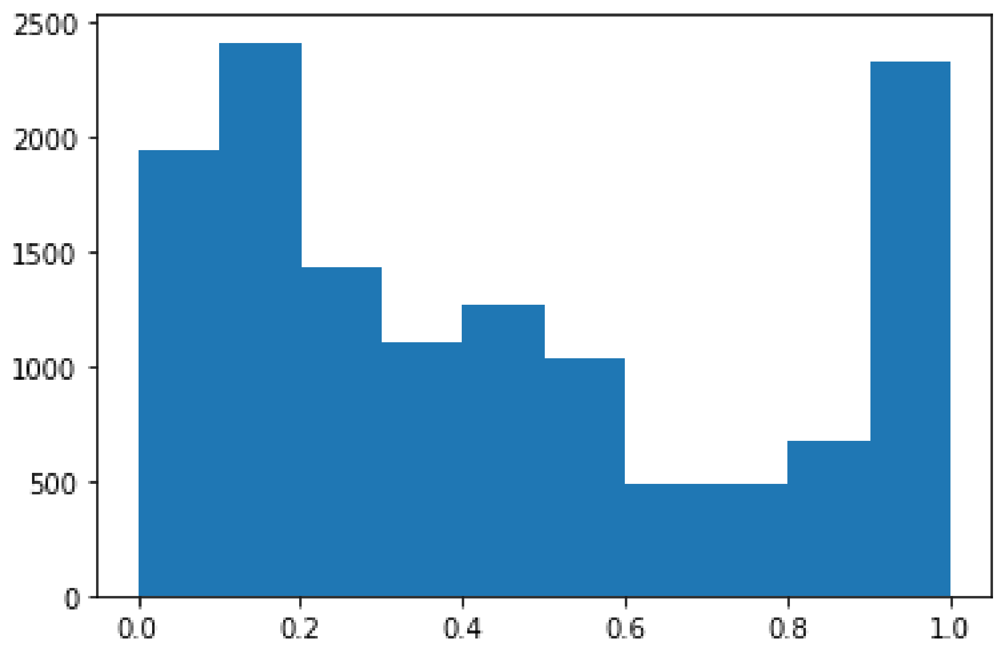


Figure 2 Distribution of the predictions for the baseline logistic regression model

After this we try a range of different learning rates, tolerances, regularization terms and thresholds. The results can be seen in table 1.

Learning rate	Tolerance	lambda	Threshold	Accuracy	F1-score	Precision	Recall
0.1	0.0001	0.0001	0.5	0.84	0.53	0.50	0.56
0.01	0.0001	0.01	0.5	0.84	0.53	0.51	0.55
0.001	0.0001	0.01	0.5	0.88	0.61	0.64	0.57
0.0001	0.0001	0.001	0.5	0.86	0.61	0.54	0.71
0.00001	0.0001	0.001	0.5	0.86	0.61	0.54	0.71
0.00001	0.0001	0.001	0.25	0.69	0.48	0.33	0.92
0.00005	0.0001	0.001	0.7	0.89	0.60	0.72	0.52
0.00005	0.0001	0.001	0.60	0.89	0.63	0.68	0.59

Table 1

We will choose the model with learning rate=0.00005, tolerance=0.0001, lambda=0.001, threshold=0.6 as the best logistic regression model. This model gives us accuracy of 0.89 and f1-score of 0.63 which is better than the baseline logistic regression model.

The cost plot and the distribution plot for this model can be seen in figure 3 and 4.

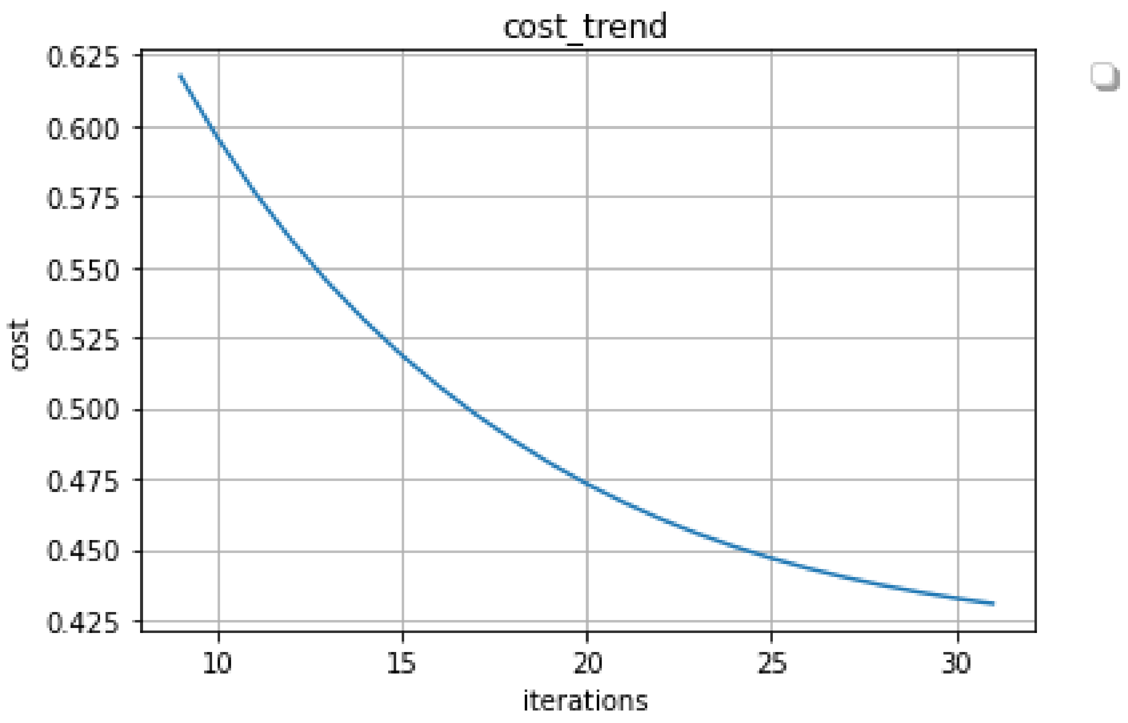


Figure 3 Cost plot for the best logistic regression model

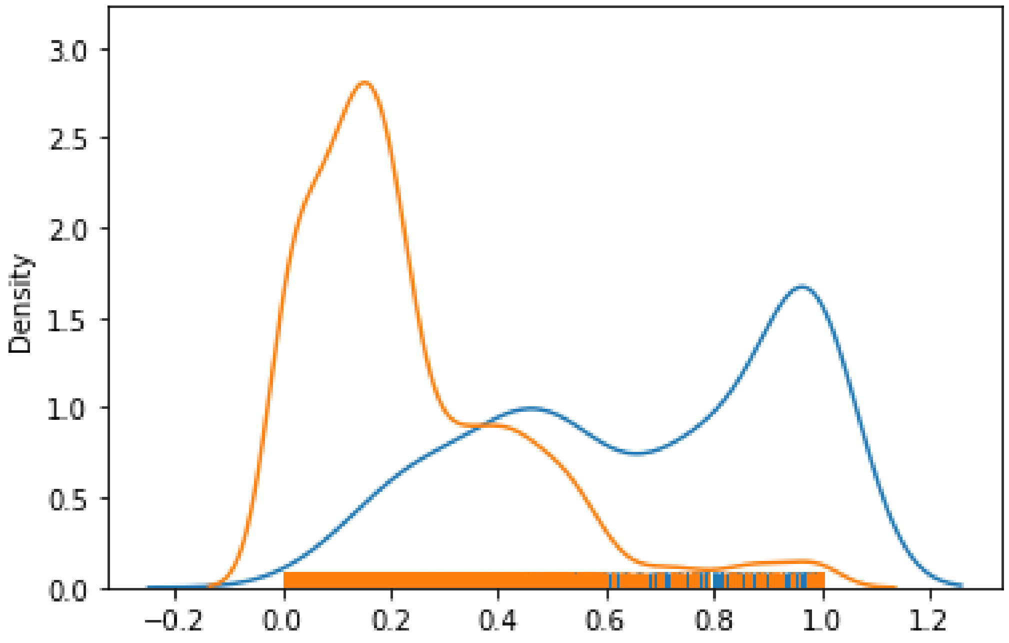


Figure 4 Distribution of the probability of predicted labels

Distributions like this will allow us to play with the threshold. For this specific task the important class for us is those who end up shopping. Therefore, we slightly increase the threshold so that we are more certain if we predict the label of an observation as True.

The precision and recall for this model are 0.68 and 0.59 respectively. This is the most balanced scores that we have for all the models.

Neural Networks

In this part we will go through different neural network models and show how to apply **bias-variance trade-off** to our models.

For neural networks we start by a simple model with 2 hidden layers and 1 output layer. We use sigmoid function for the output layer as the problem is a binary classification and for the hidden layers, we use sigmoid and Relu activation functions. The layers have 16, 32 and 1 node respectively. We go through 500 epochs for this model for a 128-batch size. We first use the oversampled dataset for training to see the results.

The results illustrate 0.88 accuracy for the train set and 0.85 accuracy for the validation set. The graph below shows the train and validation accuracy for this model. We can see that after a certain number of epochs the train accuracy starts to drift from the validation accuracy. This can be a sign that the model is overfitting. Also, the high fluctuations can show that the model is using a high learning rate. We try to reduce the number of epochs or use methods such as early stopping with specifying the patience or regularization terms to avoid overfitting. Graph 6 shows the training and validation accuracy of the same model with learning rate = 0.0001. We can see that the fluctuations have decreased.

The f1-score for the validation set is 0.63. with 0.51 and 0.81 precision and recall respectively.

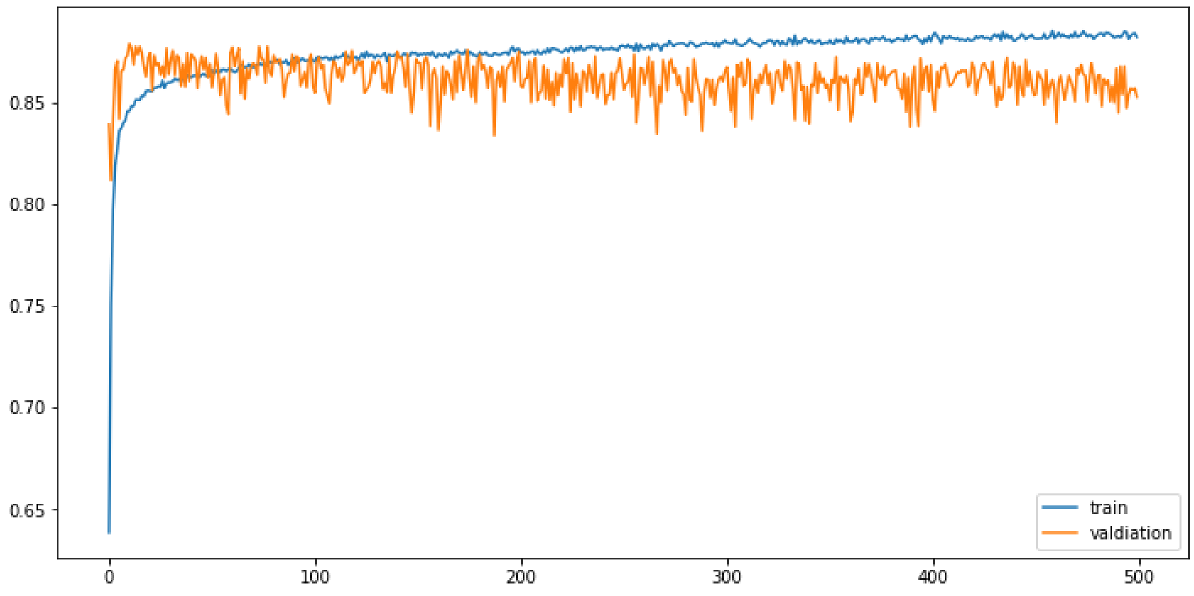


Figure 5 Train and validation accuracy for base model

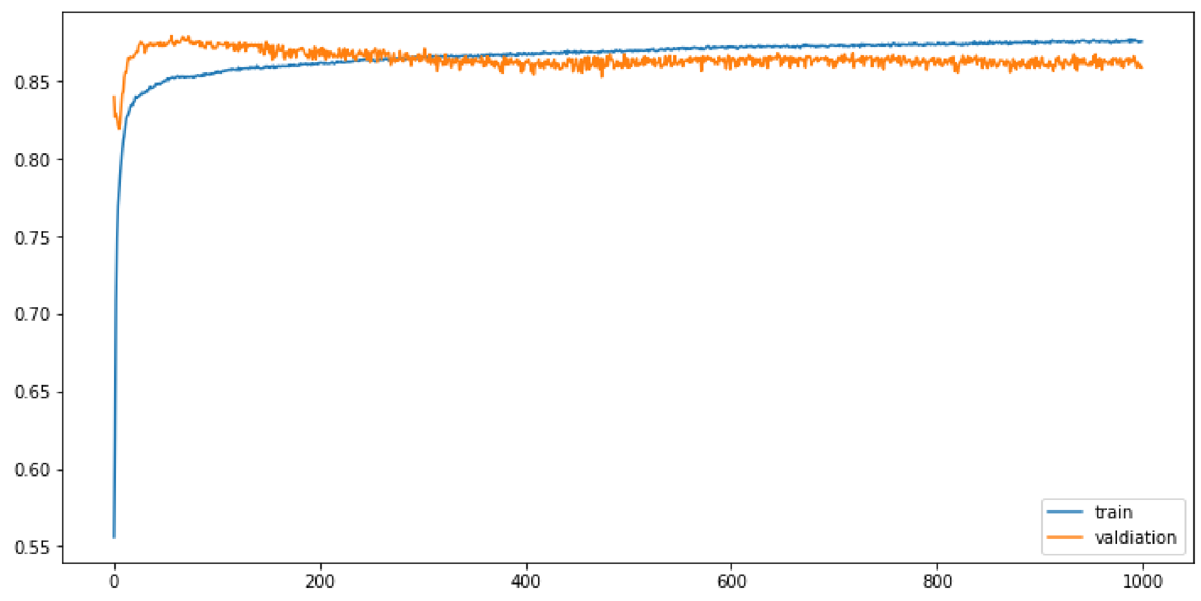


Figure 6 Train and validation accuracy for model with 0.0001 learning rate

The next graph shows the train and validation accuracy for the base model with a regularization term for the first layer. We can see that we still have the problem of high fluctuations for the high learning rate, however, the model is avoiding overfitting and the train and validation accuracies are the same.

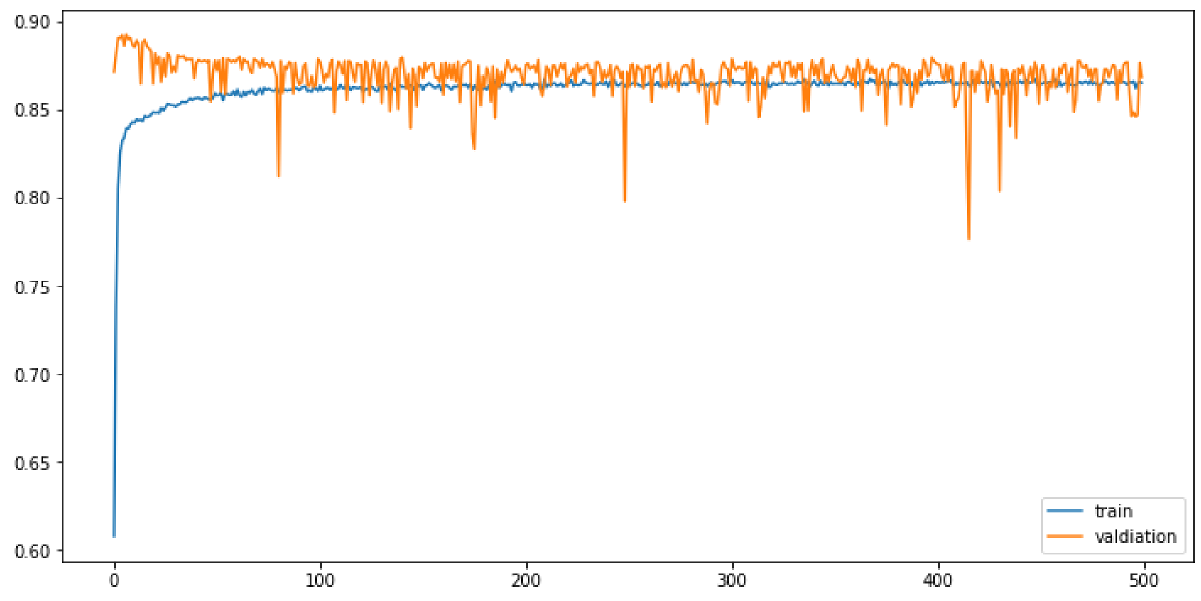


Figure 7 Train and validation accuracy for the base model with a regularization term

If we apply early stopping to our model to monitor validation loss and set patience to 5, we can see that the model only goes through 123 epochs to avoid overfitting. The reason for the higher accuracy of the validation set can be the fact that we are using oversampled data for training, but the validation set is imbalanced.

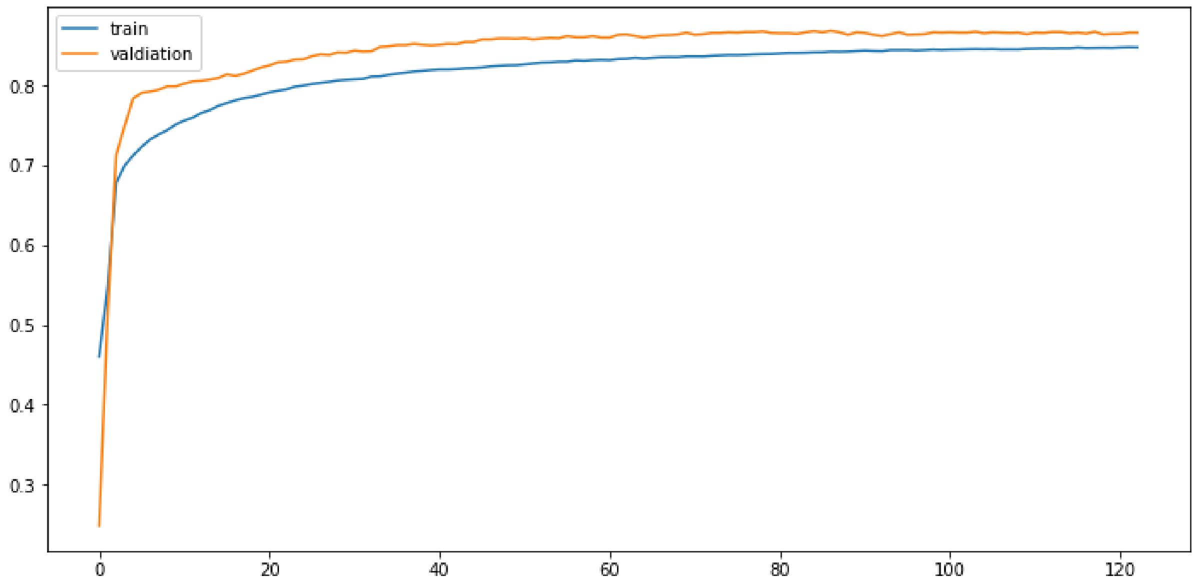


Figure 8 Train and Validation accuracy with early stopping

After examining the bias-variance trade-off, we experimented with our model with different parameters to find the best results.

Epochs	Hidden layers	Batch size	Accuracy	F1-score	precision	Recall	Threshold
200	3	64	0.86	0.65	0.55	0.79	0.5
200	3	128	0.88	0.67	0.61	0.75	0.5
200	3	128	0.88	0.67	0.62	0.73	0.6
200	3	128	0.88	0.67	0.65	0.69	0.75
400	3	128	0.87	0.64	0.55	0.77	0.5
200	4	64	0.87	0.66	0.56	0.80	0.5
200	4	128	0.87	0.66	0.58	0.76	0.5

Table 1 Neural Network Results

We chose the model with 200 epochs, 3 hidden layers and batch size of 128 as our best model. We set the threshold to 0.6 to balance the precision and recall. The train and validation accuracy and loss can be seen in the graphs.

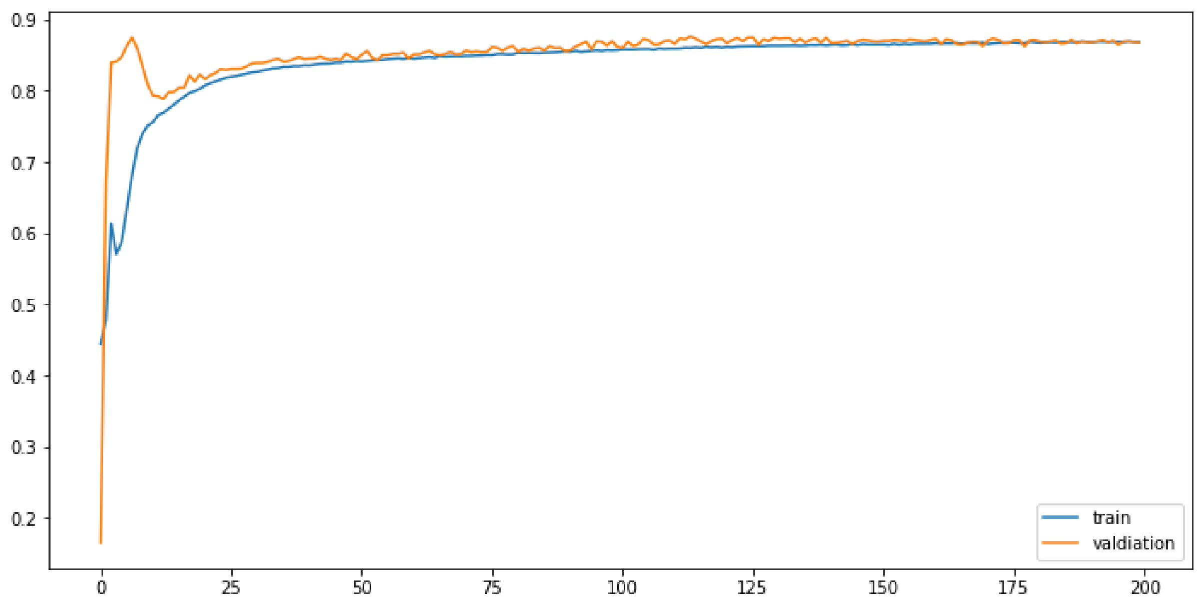


Figure 9 Train and validation accuracy

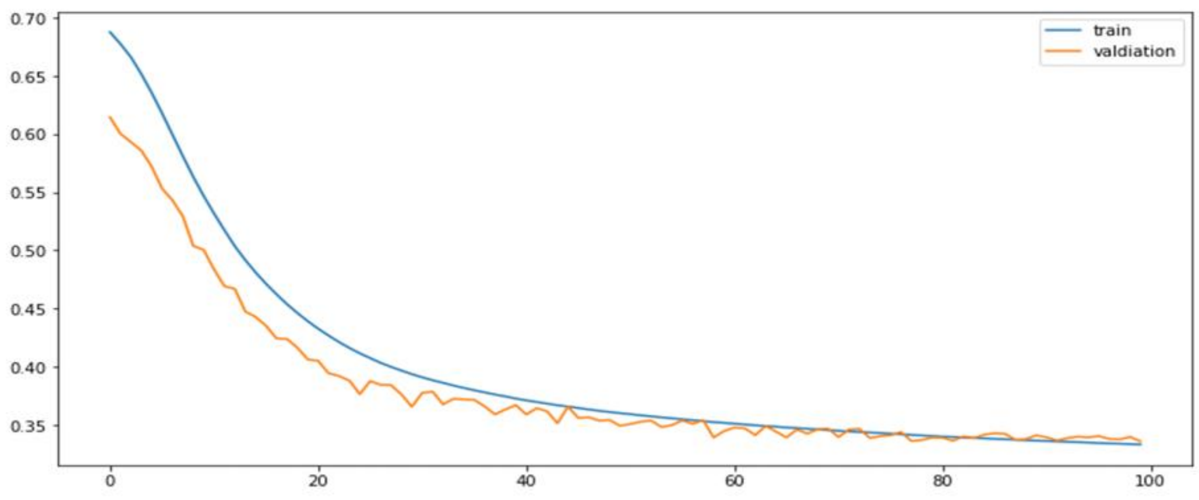
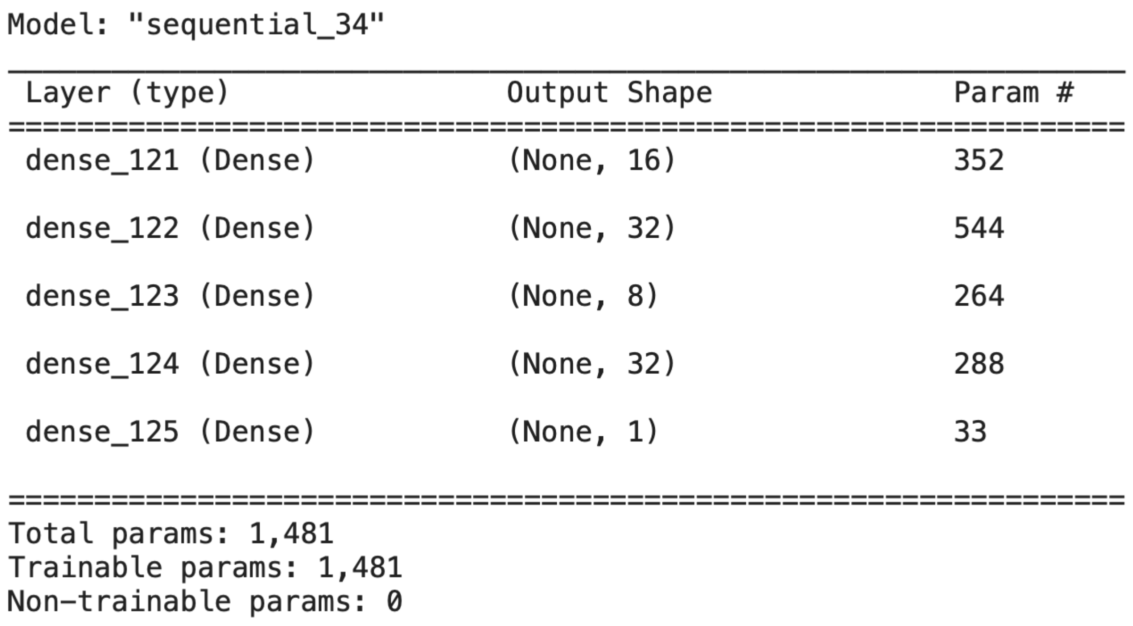


Figure 10 Train and validation loss

After plotting the results, we see the model is starting to overfit, so to prevent that we reduce the number of epochs to 100. The results for our final neural network model are seen in the table.

Epochs	Hidden layers	Batch size	Accuracy	F1-score	Precision	Recall	Threshold
100	3	128	0.87	0.67	0.62	0.74	0.6

Model summary for the 4-layer model can be seen below.



```
Model: "sequential_34"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_121 (Dense)           (None, 16)                352

 dense_122 (Dense)           (None, 32)                544

 dense_123 (Dense)           (None, 8)                 264

 dense_124 (Dense)           (None, 32)                288

 dense_125 (Dense)           (None, 1)                 33

=================================================================
Total params: 1,481
Trainable params: 1,481
Non-trainable params: 0
```

Table 2 Model Summary

Also, the distributions of the predicted probabilities can be seen in the graph below.

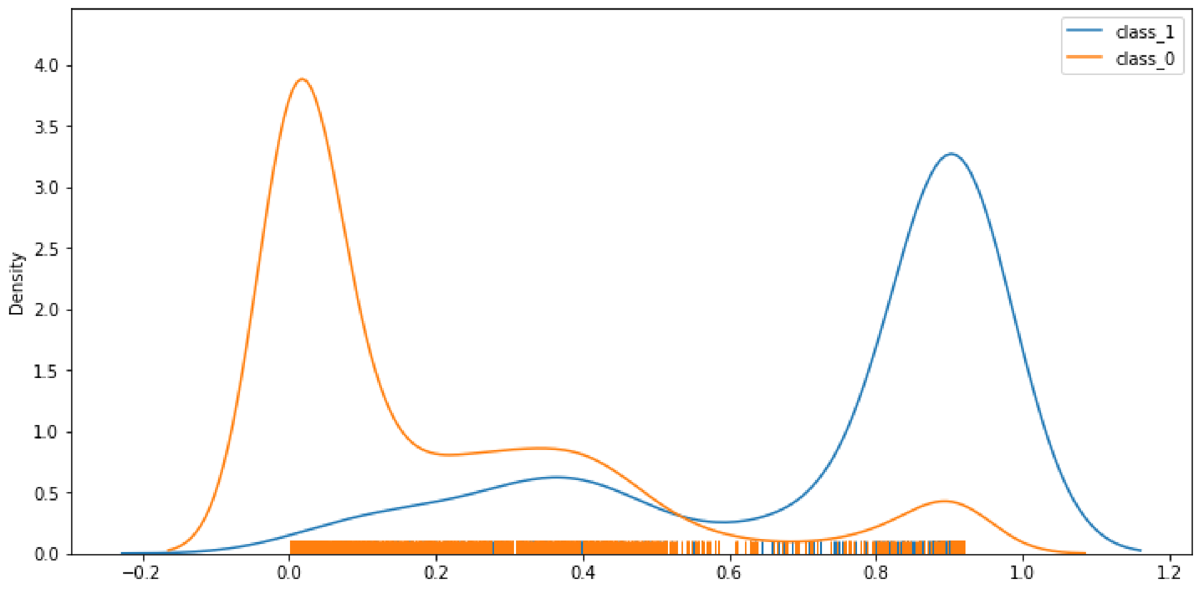


Figure 10 Predicted Probabilities

Test Set Results

After developing our models and tuning the hyperparameters, we use unseen test data to evaluate our models. The results for the 3 models can be seen in the table.

Model	Accuracy	F1-score	Precision	Recall
Logistic Regression	0.89	0.67	0.70	0.64
Naïve Bayes	0.87	0.60	0.66	0.56
Neural Networks	0.88	0.7	0.64	0.77

We can see that logistic regression and neural networks have the most balanced precision and recall and the F1-score for neural network model is the best. However, for naïve bayes the results are different. The F1-score is lower than the other two with a value of 0.60.

Discussion

In conclusion, the project discusses three models for this classification task. As there are many features for classifying customer intentions, the feature engineering part plays a pivotal role in this project. After that, we had to face the problem of imbalanced dataset. Most of the customers did end up leaving the session without shopping. For that we used SMOTE method to generate synthetic data and balance the training dataset. We must be careful to only balance the training data and keep the validation and test data imbalanced so that no outside information is leaked to the test sets. We then trained the models on the training set and chose the best models based on validation results. The best models for this task were logistic regression and fully connected neural network.

It is important that we pay attention to our goal for this project. If we want more accuracy on predicting customers who end up shopping, we can increase the threshold which will result in higher precision. Also, as covered earlier, it is important that we pay attention to bias-variance trade-off so that our model performs good for unseen data.