

Predicting Purchasing Intentions of Visitors on a E-Commerce website using Machine Learning Techniques

Prajwal Nagaraja|C21097301

In this Coursework we predicted and modelled Purchasing Intentions of the Visitors browsing an E-Commerce site using a few Machine Learning methods and algorithms. The main aim of this project is to predict if a Visitor will generate revenue i.e, purchase something from the site. To make this prediction the customer's duration on the site was recorded in a dataset. The details of the methods and the data used in this project is further explained below.

Data:

The dataset provided for this coursework had a total of 12330 rows or entries and 18 columns, each column representing a different attribute. The dataset was split into two subsections, the Numerical features(**Figure1**) section which mainly had numerical data like the 'ProductRelated_Duration' column which is the total number of time(in seconds) spent by a user reading about the different aspects and attributes of the product, and the Categorical feature section(**Figure 2**) which mainly had categorical data like the 'Region' column denoting the Geographic region from which the Visitor had started the session. The target variable or the variable that we were trying to predict for this project was the 'Revenue' column which had entries as 'True' if the visitor bought something from the website and 'False' if the visitor did not purchase anything from the website.

Numerical features

Feature name	Feature description
Administrative	Number of pages visited by the visitor about account management
Administrative duration	Total amount of time (in seconds) spent by the visitor on account management related pages
Informational	Number of pages visited by the visitor about Web site, communication and address information of the shopping site
Informational duration	Total amount of time (in seconds) spent by the visitor on informational pages
Product related	Number of pages visited by visitor about product related pages
Product related duration	Total amount of time (in seconds) spent by the visitor on product related pages
Bounce rate	Average bounce rate value of the pages visited by the visitor
Exit rate	Average exit rate value of the pages visited by the visitor
Page value	Average page value of the pages visited by the visitor
Special day	Closeness of the site visiting time to a special day

Figure1 : Numerical Features

Categorical features

Feature name	Feature description
OperatingSystems	Operating system of the visitor
Browser	Browser of the visitor
Region	Geographic region from which the session has been started by the visitor
TrafficType	Traffic source by which the visitor has arrived at the Web site (e.g., banner, SMS, direct)
VisitorType	Visitor type as "New Visitor," "Returning Visitor," and "Other"
Weekend	Boolean value indicating whether the date of the visit is weekend
Month	Month value of the visit date

Target

Revenue	Class label indicating whether the visit has been finalized with a transaction
---------	--

Figure 2 : Categorical Features and the Target Variable

Data Exploration:

Firstly we explore and inspect the dataset and get familiarised with the dataset which will be helpful in understanding the characteristics of the data. Initially we retrieve the first 10 rows of the dataset using the **.head(10)** function to check for any irregularities in the format of the dataset, then we check the number of rows and columns of the dataset using the **.shape** function which will give us an idea about the size of the dataset, next we use the **.describe()** method which will return attributes such as mean, standard deviation and the interquartile range for every numerical column present in the dataset, now we check the data types of every column using the **.dtypes** function and as our target variable can take only two values, we use the **.value_counts()** method to check if the data is balanced or not. From the **.value_counts()** method(**Figure 3**) we can see that there are 1908 entries out of 12330 entries as true which makes the dataset imbalanced. We should take this into account while measuring the performance of our models. Once we have a good understanding of our dataset we plot some of the attributes of the dataset to check for any relation or to check for outliers.

Figure 3 : Number of entries as True

```
[ ] test['Revenue'].value_counts()

False    10422
True      1908
Name: Revenue, dtype: int64
```

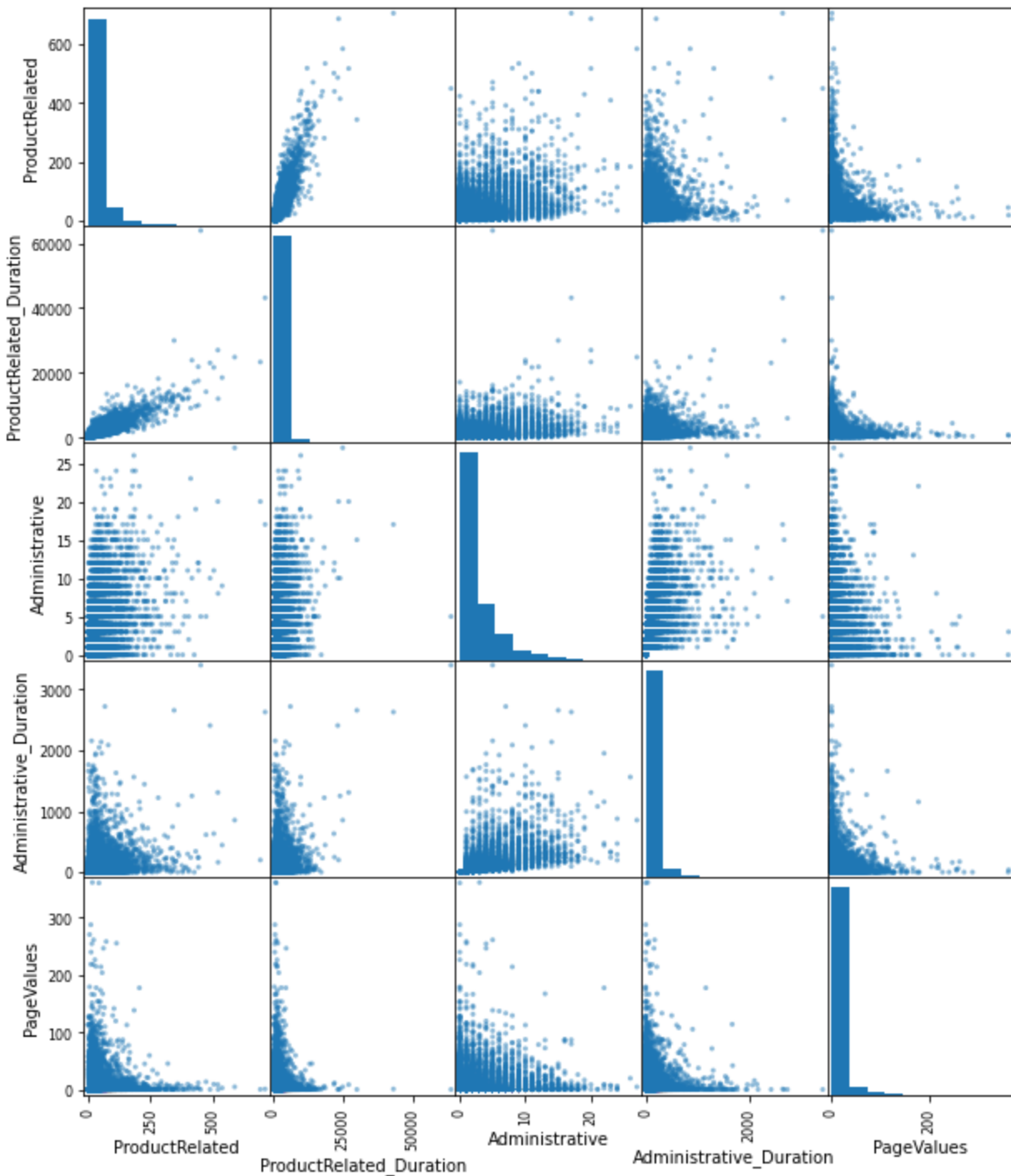


Figure 4 : Data Exploration

As seen from **figure 4**, there are some relations between the different columns of the dataset and to verify the relation between the different attributes, we use correlation heatmap(**figure 5**).

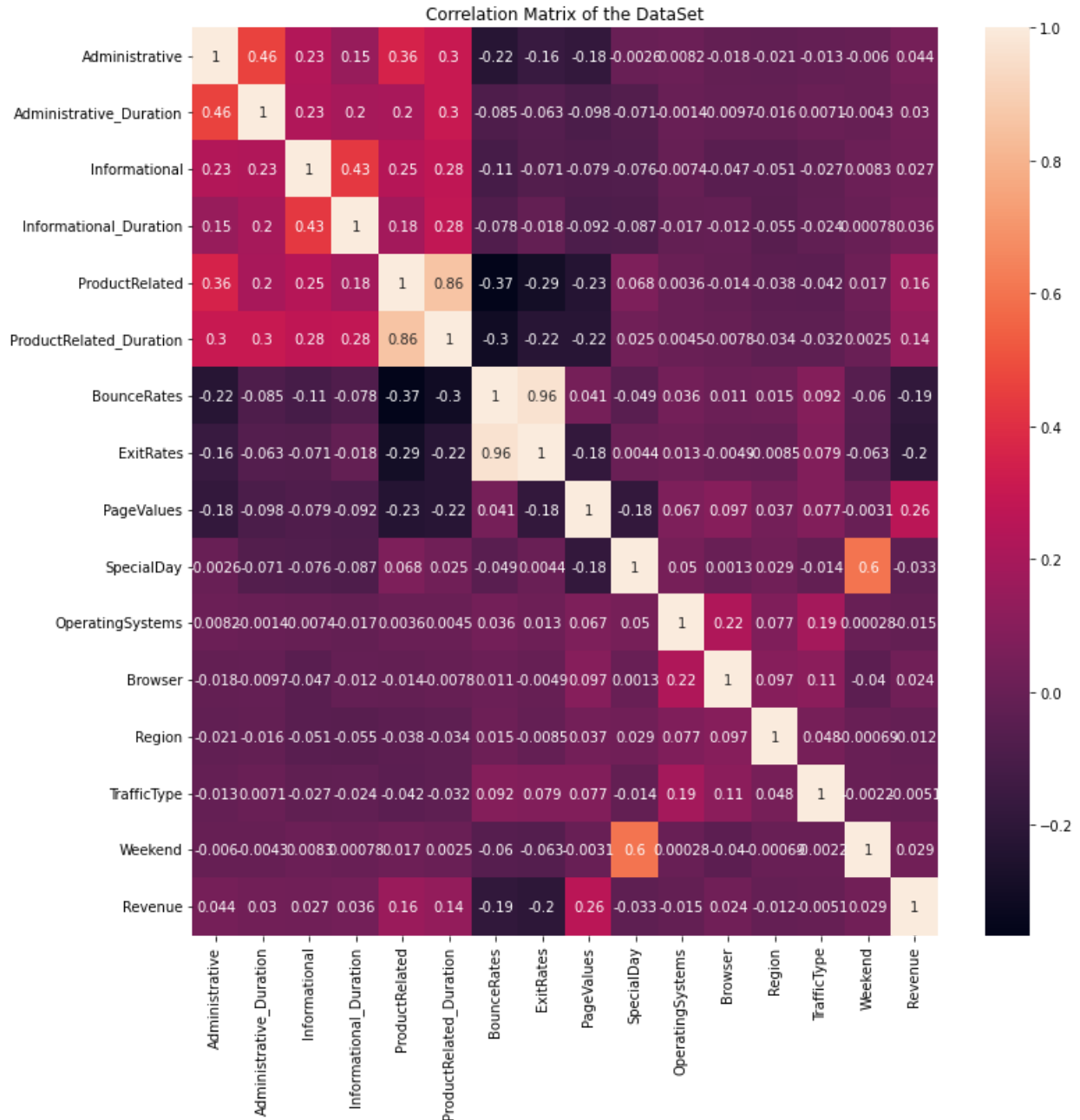


Figure 5 : Correlation Heatmap

From the correlation heatmap, we can see that our target variable 'Revenue' has significant positive correlation with only a few of the attributes. Out of these few attributes that have good correlation with 'Revenue', 'Page Values' has the highest correlation with 0.26 followed by the 'ProductRelated' and the 'ProductRelated_Duration' column with correlation of 0.16 and 0.14 respectively. Along with these 3 attributes we also include the 'Administrative' and 'Administrative_Duration' columns, although they do not have a very strong correlation with the target variable, they have significant correlation with the 'ProductRelated' and

'ProductRelated_Duration' which in turn have significant correlation with the target variable. So we only consider attributes that have significant correlation with the target variable for building the model. This ensures that the model accuracy is high because it is positively correlated to the target variable. Now that we've chosen the attributes to model our data, we plot boxplots to check for mean, standard deviation and the InterQuartile range, scatter plots to check for linear relation between the attributes and histograms which will indicate the frequency distribution for the attributes. The Individual plots for these attributes can be found on the Python Notebook.

Data Pre Processing:

Before we can start building the Machine Learning model, we have to transform and process our data into a format that is most suitable for our machine learning model. This will make it easier for our model to process the data which will result in an increase in performance and reduction in time.

Dealing with 0 entries:

The first step in pre-processing is to check if there are any missing or entries which are 0 in the dataset. To do this we replace the 0's in the dataset with NaN and use the **.isna()** method as shown in **Figure 6**.

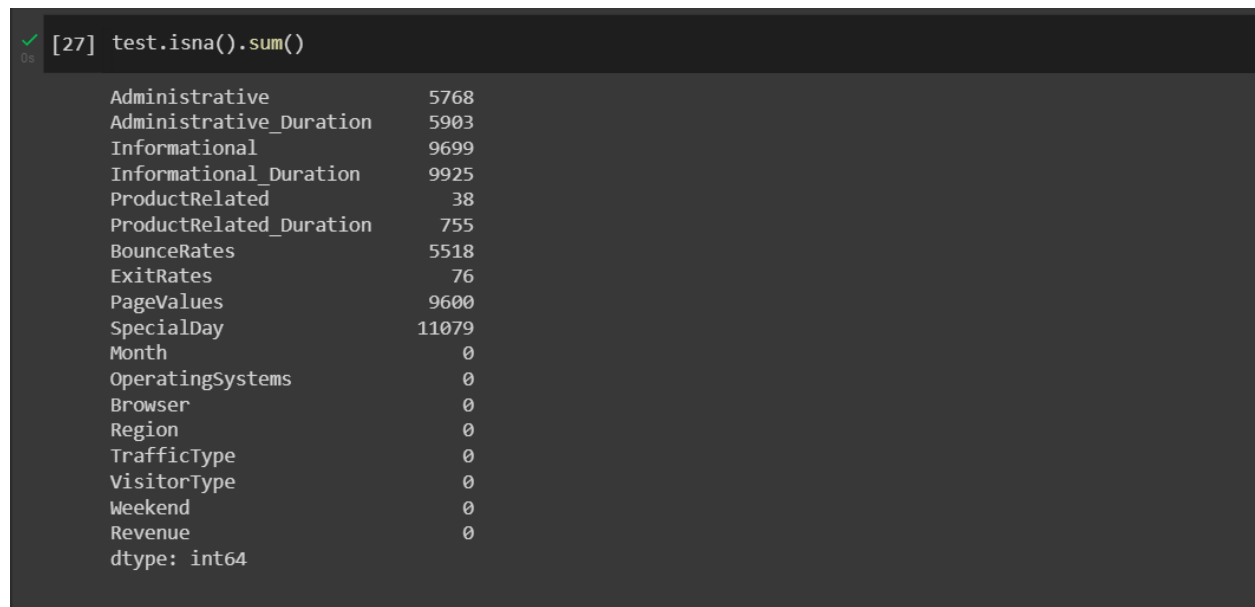


Figure 6 : Number of 0 entries in each variable.

From **Figure 6** we can see that the 'SpecialDay', 'Weekend', 'Informational' and 'Informational_Duration' columns have the majority of their entries as 0 and their correlation with the target variable 'Revenue' is quite low and negative in some

instances. So we drop these columns entirely from the dataset. Although the 'PageValues' column has the majority of its entries as 0, it has the highest correlation with the target variable so we're not going to make any changes to this column.

As we're trying to predict the purchasing intentions of a visitor, the 'OperatingSystems' and the 'Browser' columns are redundant because we're assuming that the site will function on any device irrespective of the type of operating system that the device is running on and these two columns also have low correlation with the 'Revenue' column. Hence we drop these two categorical columns entirely. As the 'ProductRelated' and 'ProductRelated_Duration' columns have significant correlation with the 'Revenue' column and only a small amount of their entries as 0 (**Figure 6**), we fill the 0 entries of the 'ProductRelated' with its mean and the 0 entries of the 'ProductRelated_Duration' with its median to ensure that the uniformity is maintained.

Encoding:

As the target variable 'Revenue' and a few other variables are categorical, we convert them to numerical values by encoding them using One Hot Encoder. Since the 'Revenue' column has only two values i.e, True or False, we need only 1 numerical value to determine if a visitor bought something from the website. The other categorical value that we encode is the visitor type which has 3 values i.e, 'New_Visitor', 'Returning_Visitor' and 'Other'. Although only 2 numbers were enough to determine the type of visitor, we use 3 numbers to make it easier to understand. After completing Encoding and dealing with 0 entry variables, we scale the data accordingly in Feature Scaling.

Feature Scaling:

Feature scaling or normalisation is the process of scaling our data to a certain range which will result in an increase in model performance. In this case we scale the 'ProductRelated_Duration' and 'Administrative_Duration' columns using minmaxscaler with range 0 to 10 because the mean and standard deviation of these two attributes is quite large and if we scale it with a maximum of 1, it would reduce the prominence of the outliers present in the data. But we scale the 'ProductRelated', 'PageValues' and the 'Administrative' columns with range 0 to 1 because the mean and the standard deviation of these two attributes are quite low and the outliers prominence is not lost while scaling. After encoding and scaling our data, it is finally ready to be implemented into our machine learning algorithm.

Model Implementation:

After Pre-Processing, Encoding and Scaling the data is converted into a format that can be easily interpreted by the computer. Even after completing all these processes, we cannot input our data directly to the algorithm. We first have to split the data into training samples which will be used to train the machine learning model and testing samples which will be used to validate our model's performance. After splitting the data, we then have to choose the type of model that we wish to build based on the target variable and since our target variable is categorical, we use classifiers to model our data. There are many classifiers that can be used to classify this dataset, but I chose Logistic Regression, Support Vector Machine and Decision Tree classifier to model the data.

Logistic Regression:

Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems. In other words Logistic Regression is a type of classifier. Although it has Regression in its name, in its basic form it uses a logistic function to model binary data i.e, it is used to classify data as either pass or fail. The regressor used in our project has the solver attribute set to 'sag' which is used to model medium to large datasets. We first split the 'ProductRelated', 'ProductRelated_Duration', 'PageValues', 'Administrative' and 'Administrative_Duration' and the target variable 'Revenue' columns into training and testing samples using the **train_test_split()** function which splits 75% of the data for training and 25% of the data for testing as shown in **Figure 7**. Now we train the model with the training samples and then validate the model by using the testing samples to predict an outcome for the test data. The performance of the model is evaluated using the accuracy metric which compares the predicted samples to the actual samples from the testing set and outputs the percentage of correct predictions for the test data as shown in **Figure 9**.

```
[117] from sklearn.model_selection import train_test_split
      X1 = test3.iloc[:, [4,5,6,7,12]].values
      y1 = test3.iloc[:,[0]].values
      X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size = 0.25, random_state =69)
```

Figure 8 : Splitting the Dataset into Training and Testing samples

```
✓ [118] from sklearn.linear_model import LogisticRegression
0s log1 = LogisticRegression(solver = 'sag', random_state = 23)
log1.fit(X_train1,y_train1.ravel())

LogisticRegression(random_state=23, solver='sag')

✓ [119] from sklearn.metrics import accuracy_score
0s pred = log1.predict(X_test1)
print(accuracy_score(y_test1, pred))

0.844905905256327
```

Figure 9 : Training and Testing the Model

From **Figure 9**, we can see that the logistic regressor has an accuracy of 84.49% i.e, it classified ~85% of the testing samples correctly. The accuracy score for this classifier is quite good but accuracy is not the only metric that we can use to measure the model performance, we will use F1- score along with the accuracy score in the results section below.

Support Vector Machine:

Another classifier that we're going to use is the Support Vector Machine. But Support Vector Machines have many hyperparameters and these hyperparameters cannot be learned and have to be chosen by us based on intuition, but we cannot be sure about the best hyperparameters to use and hence they need to be optimised to provide a better result, and as there are more than 1 hyperparameter to choose we use SVM using Gridsearch to optimise the hyperparameters which will create a grid of all the hyperparameters and verify all the combinations of the hyperparameters by testing every sample of the dataset and returning the best combination of hyperparameters along with the accuracy. And there are various kernels to choose from, and along with these kernels there are also 3 values of gamma that the model can take, along with these 2 hyperparameters there is also C which is the regularisation parameter. A combination of these 3 hyperparameters will provide the best result for our model and because this requires a lot of processing power, we only use 3 attributes that have the highest correlation i.e, we only use 'PageValues', 'ProductRelated_Duration' and the 'ProductRelated' columns to predict the target variable 'Revenue' as shown in **Figure 10**.


```
✓ [120] from sklearn.pipeline import Pipeline
      from sklearn.preprocessing import PolynomialFeatures
      from sklearn.preprocessing import StandardScaler
      from sklearn.model_selection import RepeatedStratifiedKFold
      from sklearn.model_selection import GridSearchCV
      from sklearn.svm import SVC
      X2 = test3.iloc[:, [6,7,12]].values
      y2 = test3.iloc[:, [0]].values

      parameters = {'kernel':('linear','rbf'),'gamma':[1,5,7],'C':[0.001,0.1,10]}
      model = SVC(kernel = "rbf")
      cv = RepeatedStratifiedKFold(n_splits = 7, n_repeats= 5, random_state = 4)
      search = GridSearchCV(model, parameters, scoring = 'accuracy', n_jobs=-1, cv=cv)
      result = search.fit(X2, y2)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, 1), for example using y = column_or_1d(y, warn=True)

✓ [122] print('Best Score :%s' % result.best_score_)
      print('Best Hyperparameters : %s' % result.best_params_)

Best Score :0.8606423594949102
Best Hyperparameters : {'C': 10, 'gamma': 7, 'kernel': 'rbf'}
```

Figure 10 : SVM with Gridsearch

From **Figure 10** we can see that the hyperparameters 'C' : 10, 'gamma' : 7 and 'kernel' : 7 provide the best result with an accuracy of ~86% which is slightly better than the Logistic Regression.

Decision Trees:

For our last classifier we use the Decision Tree algorithm to classify our Data. A Decision Tree is a support tool that uses a tree-like structure in which each internal node represents a "test" on an attribute and each branch from the node represents the outcome of the test. We first train the classifier with the test data that contains 75% of the entire dataset with the 'max_depth' attribute set to 10 to avoid overfitting of the data and then test the classifier with the remaining 25% of the data to predict if a visitor bought something from the site. The prediction has an accuracy of ~85% (**Figure 11**) which is on par with the Logistic Regression and the SVM. Because our data is continuous and only the target variable is categorical, the visualization of the Decision Tree will be really dense and incomprehensible as shown in **Figure 12**.

```

✓ [43] from sklearn.tree import DecisionTreeClassifier
0s tree_clf = DecisionTreeClassifier(max_depth = 10, random_state = 35)
tree_clf.fit(X_train1, y_train1.ravel())
predi = tree_clf.predict(X_test1)
print(accuracy_score(y_test1, predi))

```

0.844905905256327

Figure 11 : Decision Tree Classifier

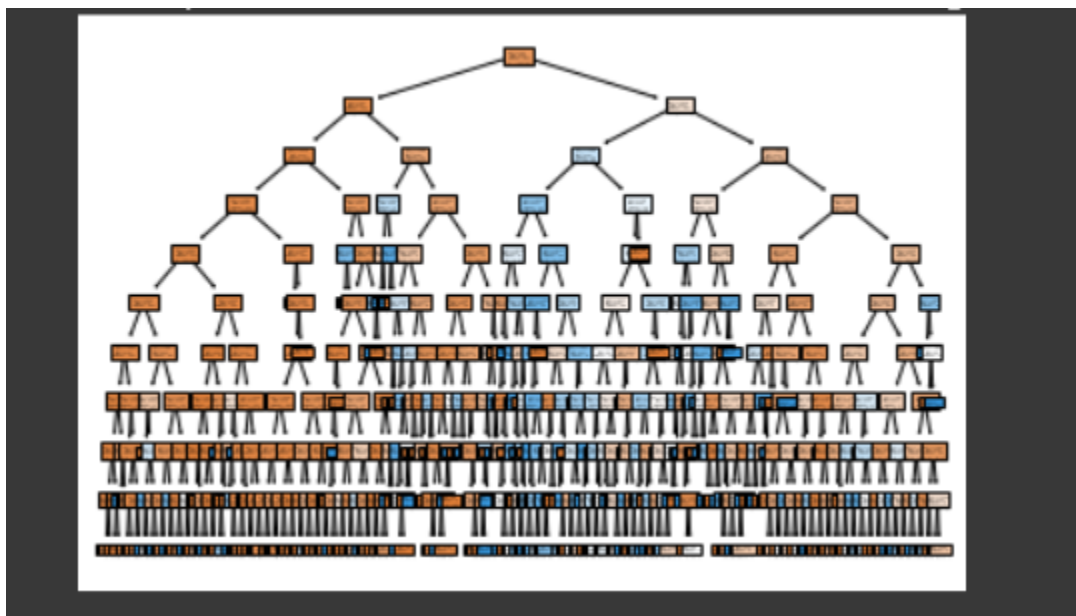


Figure 12 : Visualisation of Decision Tree

To improve the performance of the Decision Tree model and reduce variance of the data, we use Decision Tree Classifier with Bagging(Bootstrap Aggregation) which creates several subsets of the of the training data with replacement and each subset is used to train the classifier and as a result we end with an ensemble of different models and the average of all the trees is used to make the decision which is more robust and will aid in the reduction of variance. For our model we use 500 trees to train 1000 bootstrapped subsets of the dataset with replacement as shown in **Figure 13**, and because of this method there is a slight increase in the accuracy to ~86% as compared to a normal decision tree with an accuracy of ~84%(**Figure 13**).

```
✓ [45] from sklearn.ensemble import BaggingClassifier
3s      from sklearn.tree import DecisionTreeClassifier, plot_tree

      bag = BaggingClassifier(
          DecisionTreeClassifier(random_state = 2), n_estimators = 500,
          max_samples = 1000, bootstrap=True, random_state = 69)
      bag.fit(X_train1, y_train1.ravel())
      y_preddt = bag.predict(X_test1)

✓ 0s      print(accuracy_score(y_test1, y_preddt))

      0.8627514600908501
```

Figure 13 : Decision Tree Classifier with Bagging

Results and Conclusion:

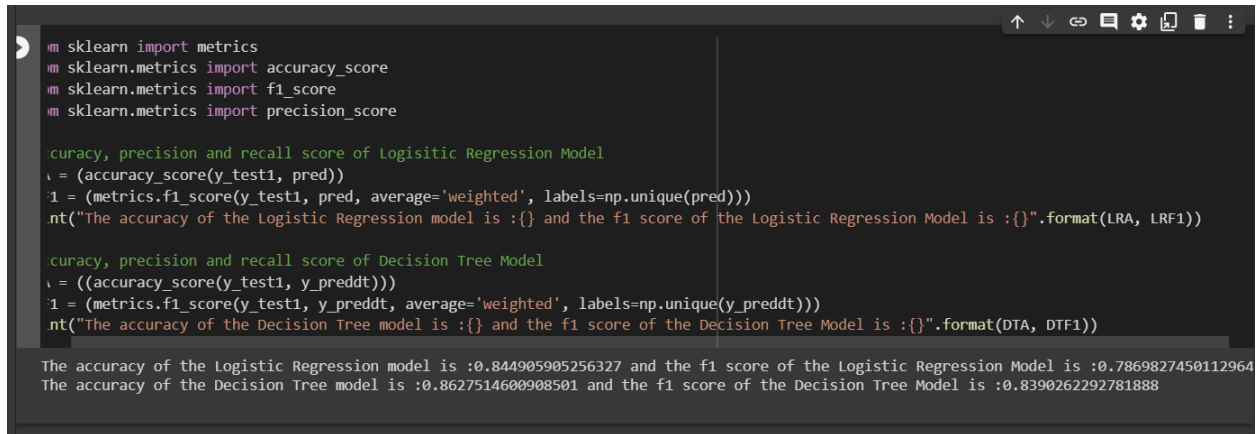
From the above results we can see that the accuracy of all the three models is between 84% and 87% and accuracy for a classification model is the fraction of samples that the model predicted correctly. So in our case all the 3 models were able to successfully predict at least 84% of the test data correctly. But accuracy for this imbalanced dataset is not a good measure of performance, because the Dataset has only about 1908 entries out of 12330 entries marked as True i.e, only 15% of the customers bought something from the site and hence the accuracy of our models is the models predicting the entries that are marked as false. Hence we also use F1 score which is a combination of precision and recall as shown in **Figure 14** to check our model's performance.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Figure 14 : F1 score

And in our case we use the F1_score attribute average set to 'weighted' which highlights the importance of the entries that have generated revenue and because F1 score is a combination of Precision and Recall, a good F1 score implies that the model has good performance. From Figure 16, we can see that the F1 score for the Logistic Regression model is 0.78 and the F1 score for the Decision Tree Classifier is 0.83

which is quite high. This implies that both the models performance is really good and this also implies that the F1 score of the SVM will also be around the scores of these 2 models.



```
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score

# Accuracy, precision and recall score of Logistic Regression Model
a = (accuracy_score(y_test1, pred))
f1 = (metrics.f1_score(y_test1, pred, average='weighted', labels=np.unique(pred)))
print("The accuracy of the Logistic Regression model is :{} and the f1 score of the Logistic Regression Model is :{}".format(LRA, LRF1))

# Accuracy, precision and recall score of Decision Tree Model
a = (accuracy_score(y_test1, y_preddt))
f1 = (metrics.f1_score(y_test1, y_preddt, average='weighted', labels=np.unique(y_preddt)))
print("The accuracy of the Decision Tree model is :{} and the f1 score of the Decision Tree Model is :{}".format(DTA, DTF1))
```

The accuracy of the Logistic Regression model is :0.844905905256327 and the f1 score of the Logistic Regression Model is :0.7869827450112964
The accuracy of the Decision Tree model is :0.8627514600908501 and the f1 score of the Decision Tree Model is :0.8390262292781888

Figure 15 : Accuracy and F1 score of the two models

With this we can conclude that all the three models have good performance and are able to predict the purchasing intentions of the visitor for that site with an accuracy of ~80%.