Applied Machine Learning (CMT307)

student number: C1986330

Question 1

	PREDICTED CLASS		
ACTUAL CLASS		**Class=True**	**Class=False**
	Class=True (positive)	**TP = 7**	**FN = 2**
	Class=False (negative)	**FP = 4**	**TN = 7**

Precision = True positives / (True positives + False positives)

P = 7/ (7+4) = 7/11 ≈ **0.636**

Recall = True positives / (True positives + False negatives)

R =7/ (7+2) = 7/9 ≈ **0.77**

F–measure = (2* Precision* Recall) / (Precision + Recall)

F = (2*0.636*0.77)/ (0.636+0.77) = 0.989/1.416 ≈ **0.698**

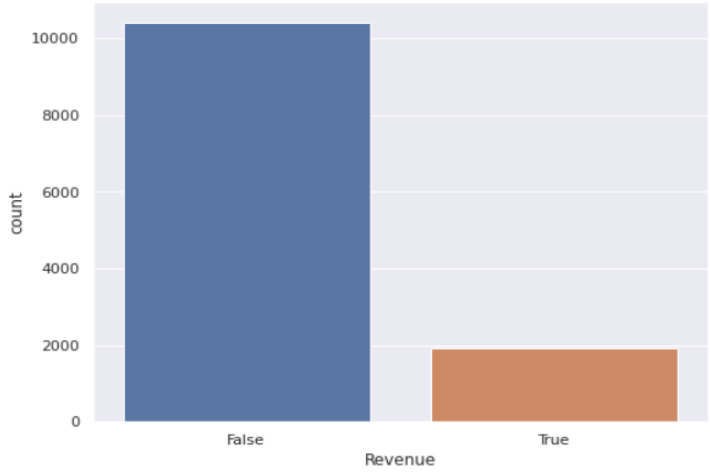
Accuracy = (True positives + True negatives)/ (True positives + False negatives+ True negatives + False positives)

= (7+7)/ (7+2+7+4) = 14/20 = **0.7**

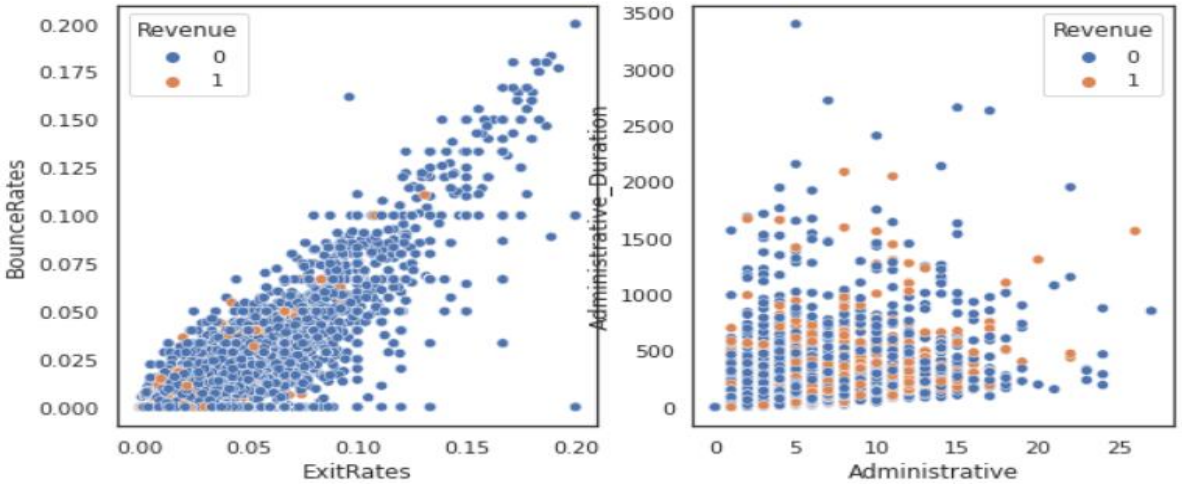
Question 2

Data exploration

–The dataset consists of 12,330 rows of online shopping visits to a website, and 18 features describe each row, divided into numeric and categorical features. Feature '**Revenue'** is the **label** of our binary classification problem. Our goal is to predict the label 'Revenue' to see whether a visit session will end up with a transaction. After plotting the revenue, we can see that a few visitors make a purchase. We suspect that the data is **imbalanced**, as shown below.



In this section, we used multiple plotting types such as a histogram, scatter and box plot to gain familiarity with the features. The distributions of the values in the individual features indicated which features correlate with each other. Furthermore, outliers can be clearly seen on the box plot. These are correlated features with the target, as examples of what we have done on the attached Python file.



Data pre-processing

Since the dataset is not missing values, we started splitting the data into various train and test data sets. The training dataset (x_train) will use 80% of the data. The test dataset (y_test) will use the remaining 20% of the data.

The datasets y_train and y_test contain the respective prediction labels. Subsequently, we encoded categorical features using OneHotEncoder and OrdinalEncoder from sklearn for Visitor Type and Month, respectively.

Whisker plots had shown many outliers, so to detect them, we used IQR. Some features had more than 2,000 outliers, so we decided to keep them, as they may have essential values. This provides the right candidate for using a robust scaler transform to standardise the data in the presence of skewed distributions and outliers.

The number of outliers was detected using IQR.

```
#Detect outliers using IQR.
((Outliers < (Q1 - 1.5 * IQR)) | (Outliers > (Q3 + 1.5 * IQR))).sum()
```

```
Administrative             404
Administrative_Duration   1172
Informational              2631
Informational_Duration     2405
ProductRelated              987
ProductRelated_Duration     961
BounceRates                1551
ExitRates                  1099
PageValues                 2730
dtype: int64
```

As mentioned, we are dealing with an imbalanced dataset, which means one class is dominating. There are limited instances of the other type. To treat that, we chose to increase minority class instances by using oversampling, as it helps retain the data's variability. The SMOTE algorithm (Synthetic Minority Oversampling Technique) was implemented.

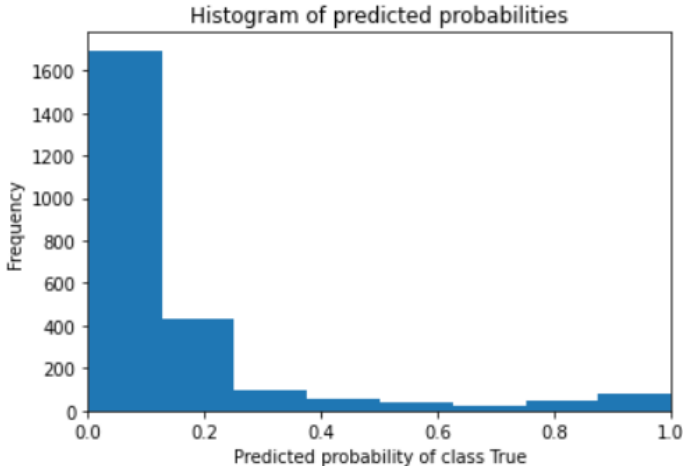
Model implementation

First, we started by testing many classifiers used for a classification project, including Logistic Regression, K-Nearest Neighbors (KNN), SVM, Kernel SVM, Decision Tree classification and RandomForest. Most of them have similar performance, so we will explain three, according to what was asked in the question.

The XGBoost algorithm was suitable for a classification predictive modelling problem because it provides a set of hyperparameters that control the model training procedure. However, the algorithm works fine by default on our processed data set. We have tried it on imbalanced classification datasets because it provides a way to fine tune the training algorithm to pay more attention to the minority class misclassification of datasets with skewed class distributions. This modified version of XGBoost, called a *Class Weighted XGBoost*, could better perform binary classification problems with severe class disruption. Moreover, it is possible to achieve better performance by weighting a different class, but we were more interested in the positive class. A grid search was used to determine the best parameters.

As we work with an imbalanced dataset, we implemented logistic regression because it might refer to probabilistic algorithms, as they often fit under a probabilistic framework. Good accuracy on the training set and the test set was obtained by default, but the recall and precision were low for the positive label. This was achieved using a threshold, such as 0.5, where all values equal to or greater than the threshold are mapped to one class, and all other values are mapped to another class. The cost of the true type of misclassification is more important than another type of misclassification.

As shown below, most of the true class probabilities are less than the default threshold (0.5), so we decreased the threshold for predicting true to increase the classifier's sensitivity.



K-NN was implemented, but it did not perform as well as other algorithms regarding recall or precision, although we searched for the optimal value for K using cross validation and a grid search. Therefore, we prefer to use Kernel SVM because it can work well with many features and is suitable for classifying extreme binary states with fewer outlier effects.

Performance evaluation

A comprehensive and clean way to illustrate the results of the classification model is the Confusion Matrix. It differentiates between expected and actual designations. Besides, to evaluate the models' performance, we used standard evaluation metrics for classification problems such as recall, precision and f1 score for both labels, and the easiest way to track them with each model or when changing parameters is to print a classification report.

Reports of the best performance models

This is for XGBClassifier. The first line represents the accuracy in the test set, and the matrix is FP TN, then FN TP.

```
0.8929440389294404
[[1973   82]
 [ 182  229]]
              precision    recall  f1-score   support

       False       0.92      0.96      0.94      2055
        True       0.74      0.56      0.63       411

    accuracy                           0.89      2466
   macro avg       0.83      0.76      0.79      2466
weighted avg       0.89      0.89      0.89      2466
```

Logistic regression classification report

```
[[1876  179]
 [ 133  278]]
              precision    recall  f1-score   support

       False       0.93      0.91      0.92      2055
        True       0.61      0.68      0.64       411

    accuracy                           0.87      2466
   macro avg       0.77      0.79      0.78      2466
weighted avg       0.88      0.87      0.88      2466
```

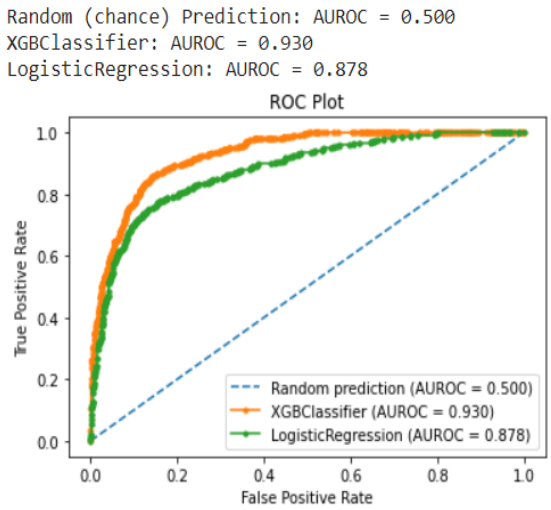
The K-NN and SVM kernels did not perform as expected at this stage of the evaluation, but as we mentioned before, cross-validation, grid search and random search were used during implementation. At this point, we defined features using the SelectKBest function and dropped the less critical features, but the models did not show a considerable difference from the first performance.

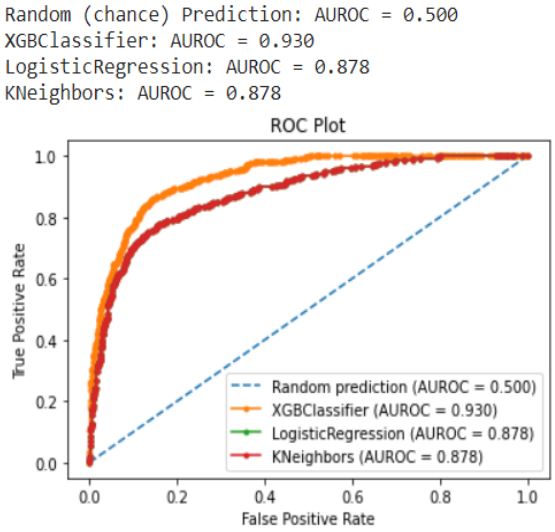
<u>The most important features</u>

```
print(featureScores.nlargest(10,'Score'))  #print 10 best features
```

```
                  features          Score
5   ProductRelated_Duration  877404.339415
8               PageValues  175126.808512
1   Administrative_Duration   41754.836841
3    Informational_Duration   35059.775770
4           ProductRelated   19317.285376
0           Administrative    1133.965531
2            Informational     357.981605
10                   Month      86.163696
9              SpecialDay      53.797094
15            VisitorType      37.547523
```

ROC curve for the three models, logistic regression and k-nn have the same performance.

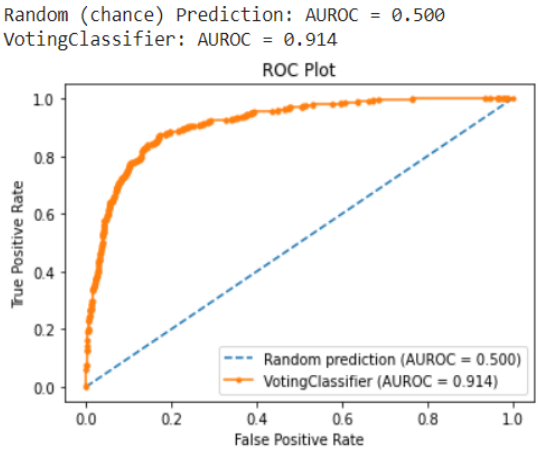




Result analysis and discussion

We have developed a classification model for predicting the purchase intentions of online shoppers. Namely, we have trained and tested an XGboost classifier, logistic regression, K-NN and SVM kernel algorithms. Hence, as described in the ROC_curve, it is one of the most common measures for evaluating machine learning algorithms' performance, especially when working with imbalanced data sets. All models function similarly, but the best one was the XGboost classifier. The plots above clearly show that, even when combined by VotingClassifier with all three models, XCboost is still the top performer.

ROC curve for VotingClassifier



References

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition by Aurélien Géron.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT Press,2016.

Imbalanced Classification with Python, Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning by Jason Brownlee, (e-book).