



UCD Michael Smurfit  
Graduate Business School








## Machine Learning for Finance



## Bank telemarketing and machine learning

## Group Project Submission

Word Count: 2663

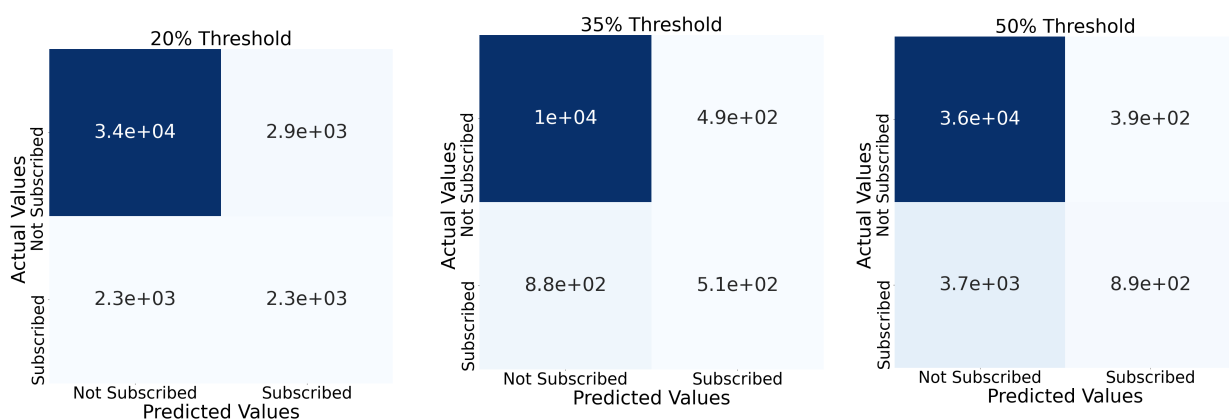
Student Name	Student Number
Ross Kearney	
Fred 	
Brian 	
Mark 	

## Question 1

(a) Before fitting algorithms to the training set it is important to establish some context for the purpose of the algorithm. The algorithm will be used to predict whether or not a customer will subscribe to a term deposit in a banking institution. Therefore, a model which has too many false positives will cause damage to the bank as they may spend large amounts on targeting these false positive customers with advertising or extra services and may lead the institution to believe that they will have a larger customer base available in the future. On the other hand, a model with a high true positive rate will improve the efficiency with which the bank can target prospective term deposit customers. We suggest that the bank will be seeking a model with high predictive accuracy and a high number of true positives.

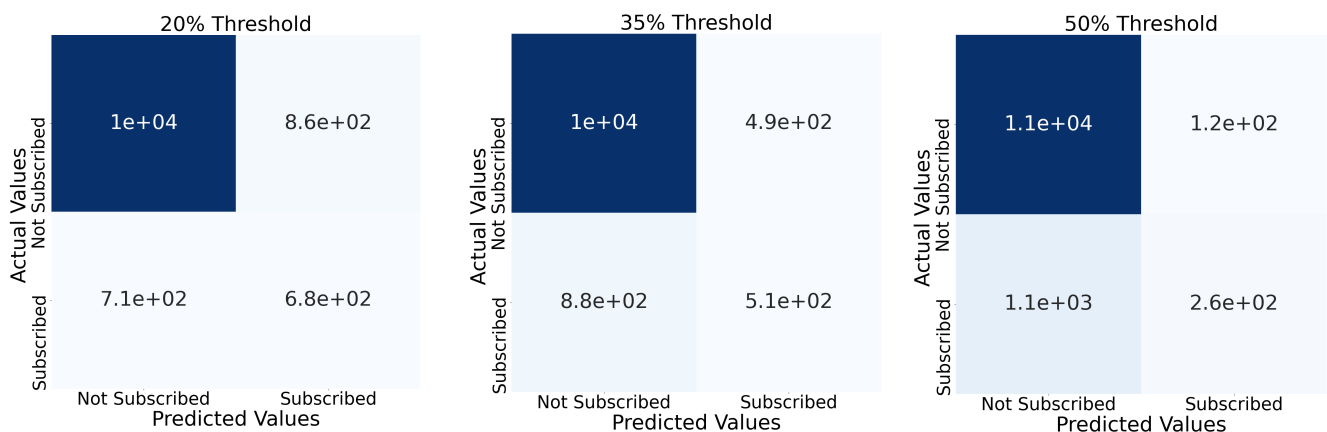
After inputting data, we first note that many of the features contain categorical data, rather than numerical or binary information. So, we decide to code the variables job, marital, education, default, housing, loan, month, day\_of\_week, poutcome and contact as dummy variables, meaning for each of the columns, we create new columns equal to the number of possible values in the category minus one, lengthening the dataset to fifty-five columns.

Next we note that the duration column may disimprove predictions, due to the fact that duration is zero, when the term deposit is zero. After treating the data, we now fit a logistic regression from the Scikit-Learn library on the data and use it to predict the probability that an observation is actually in the term deposit = 1 class. We use these probability predictions and threshold values to assign observations in the term deposit class. For example, in the first case, we predict observations as belonging to the term deposit = 1 class, if the predicted probability is equal to or above 20%. We then use these to plot confusion matrices.

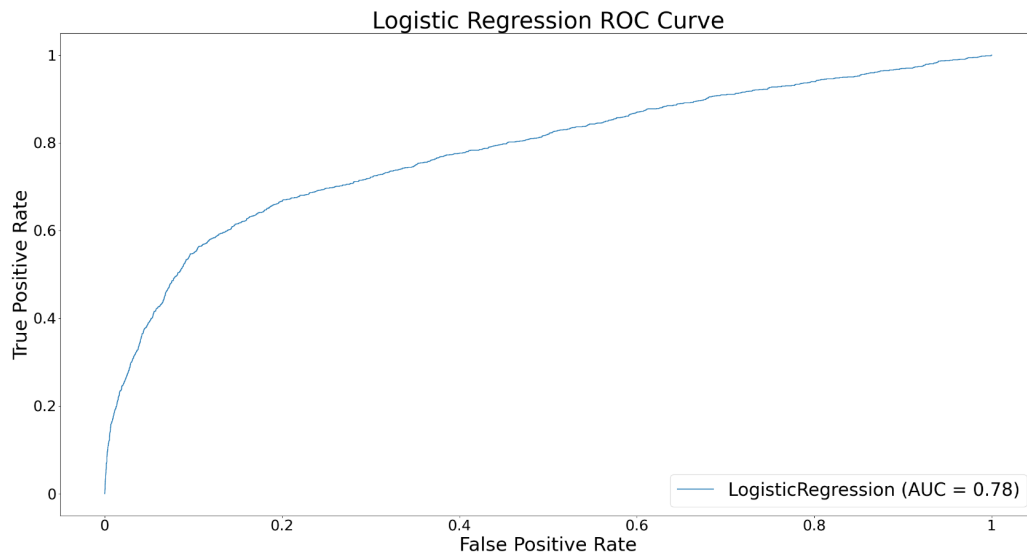


From above, we can calculate the true positive and false negative rates for each of the different threshold values. From left to right, we get respective true positive rates of 49.5%, 36.7% and 19.2% and false positive rates of 7.9%, 4.5% and 1.1%. A high true positive rate and a low false positive rate is most favourable, however, as mentioned above, we have decided to pick true positive rate as the more important metric. So, even though the model with a 20% threshold probability has a higher false positive rate than the others, it also has the highest true positive rate, making it the preferred model.

**(b)** Next, we divide the dataset into training and test sets, with 30% going into the test set. Then, by fitting models to the training set and testing their predictions on the training set, we can get a more accurate representation of how well the model will perform in the real world, on out-of-sample data. Below are the test set confusion matrices for these models.



Here, calculating the true positive rates, we have 49.0%, 36.5% and 19.1%. Respective false positive rates are 7.8%, 4.5% and 1.1%. So, we can see that there is no change in the relative performance of any of the models in these metrics, meaning that we again choose the model with a 20% threshold as the preferred model. Accuracies are 87.3%, 88.9% and 89.9% respectively.

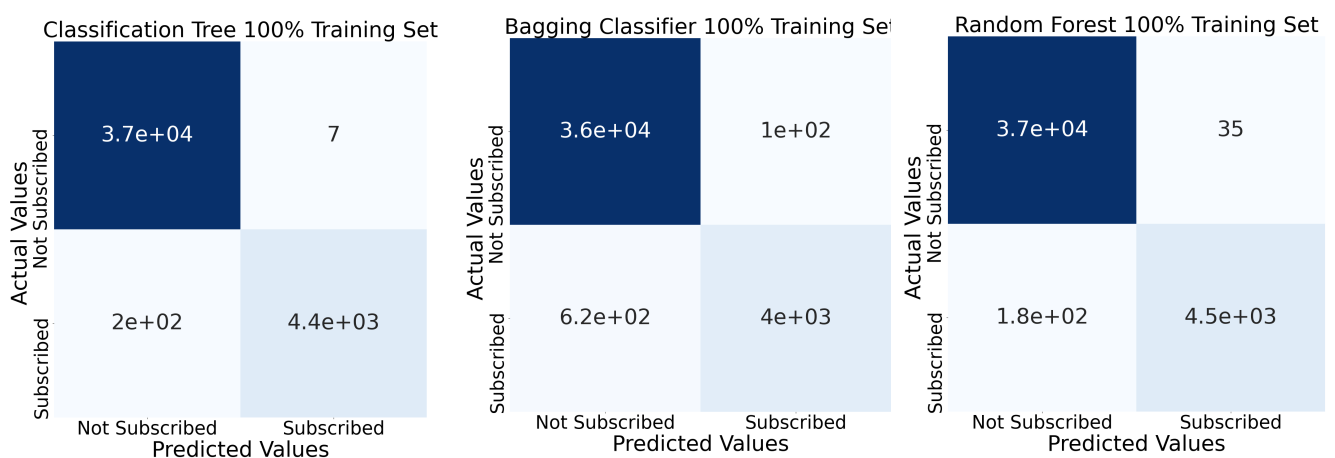


(c) Above we have the ROC curve and corresponding AUC for the logistic regression model on the test dataset. The receiver operating characteristic curve plots the true positive and false positive rates for a model while varying the threshold probability, as we have done manually above. A perfect model would be just a point where the false positive rate is zero and the true positive rate is one for every threshold probability and a poor model with no predictive power over flipping a coin would be a  $45^\circ$  line from (0,0) to (1,1). The area under the curve is a way of distilling the information in this graph into a single number, where 1 would be a perfect model and 0.5 would be a model that is no better than guesswork. Therefore, both the ROC and an AUC of 0.78 confirm that this model is quite a good predictive model, but not perfect.

## Question 2

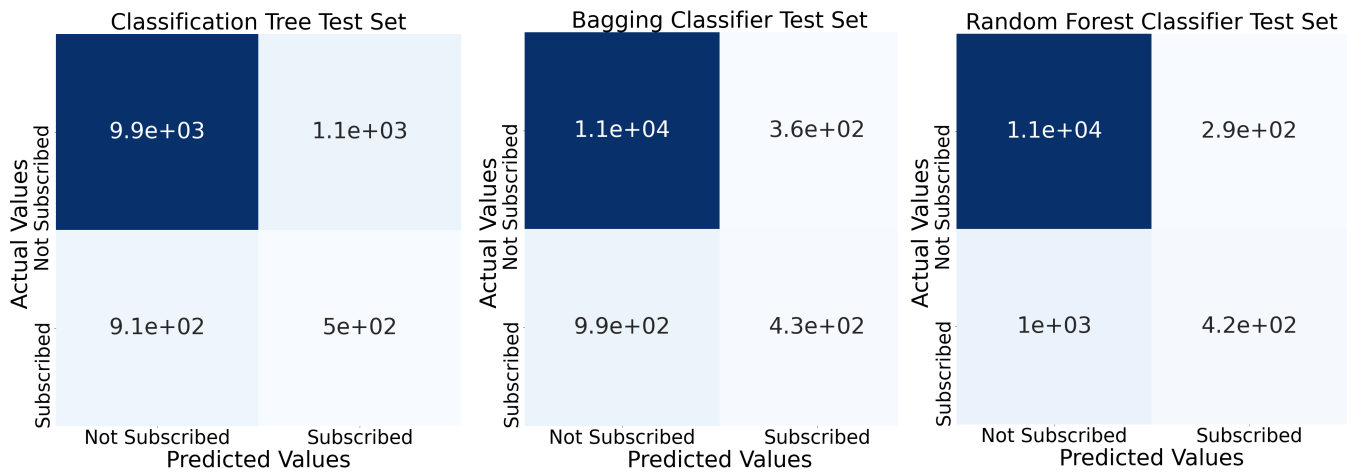
(a) Firstly, a classification tree, bagging and random forest models were fit on the entire dataset. This is poor practice when fitting a machine learning model as it results in the model being overfit to the training data and leads to poor out of sample performance.

When examining the predictive accuracy of the fitted models over the entire set they show near 100% accuracy as the test set is the training set meaning the models have already been fit to all of these observations. Examining the confusion matrices shows the overfitting problem.



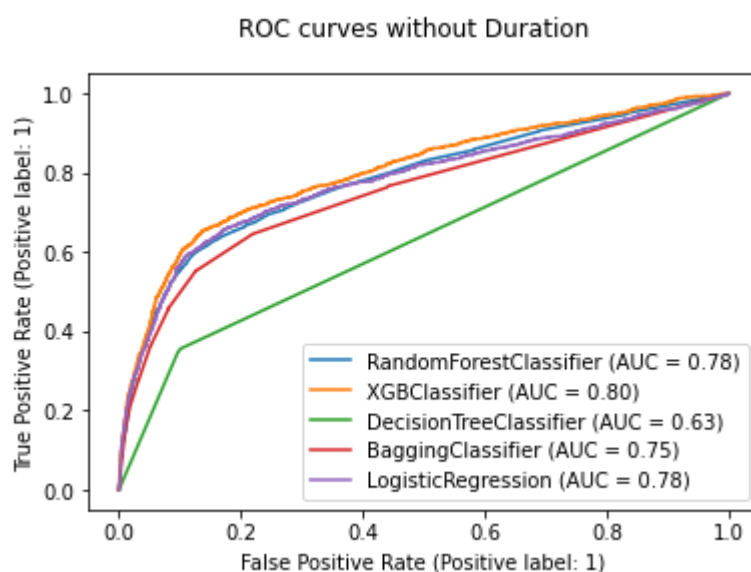
As can be seen from the above figure the predictive accuracy for all three models is extremely high (99.5%, 98.3% and 99.5% respectively). The respective true positive rates are 78.1%, 86.6% and 96.1% and the false positive rates are 2.8%, 0.3% and 0.1%. So, we can see that in all measures, these models are an improvement over the logistic regression model, when fit and tested on the entire dataset.

(b) A way to avoid overfitting is to split the data into a training set and a testing set. This was conducted subsequently with the model being fit on 70% of the data and the remaining 30% being withheld to evaluate the predictive accuracy of the models on unseen data.



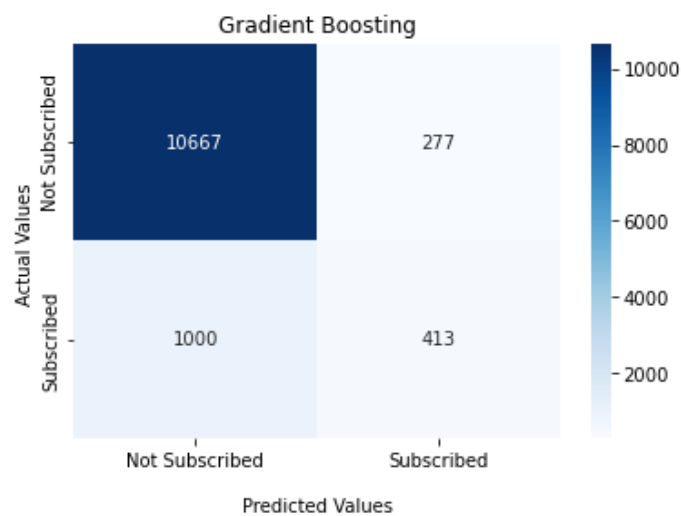
The models in this example perform relatively well achieving predictive accuracies of 84.2%, 89.0% and 89.5% on the unseen data points. This model was trained on only 70% of the data and was able to accurately predict the term deposit class among the remaining data. An important thing to note however is the imbalance in the data in the training set and test set, there are far more customers in the not subscribed class than in the subscribed class, so a model which predicts term deposit = 0 will have high accuracy. Looking at the true positive rates of 34.7%, 28.4% and 29.5% and false positive rates of 9.8%, 3.3% and 2.7% show this, since the false positive rates are quite low, but the true positive rates are not very impressive.

Below, we plot the ROC curves for these models, as well as an XGBoost model, which is also an ensemble tree method, as are the bagging and random forest models.

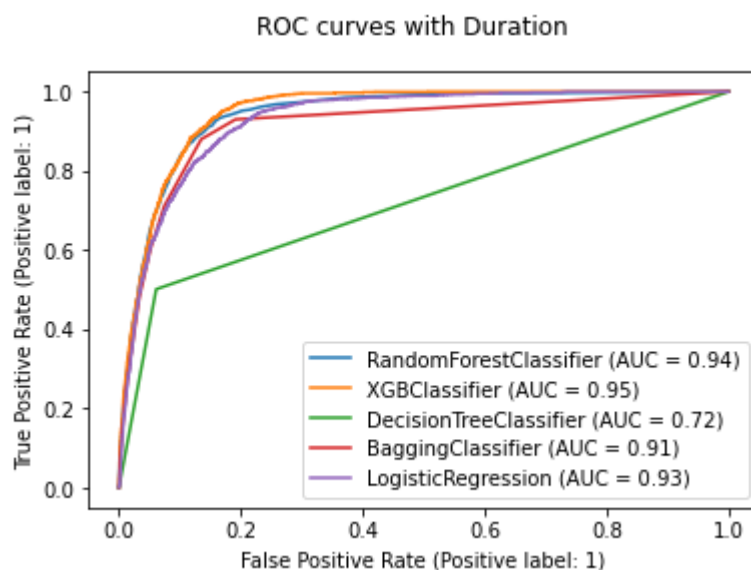


The ROC curve was sketched for each of the following models, random forest, gradient boosting, the decision tree, bagging and the logistic regression from question 1. The AUC is a good metric for evaluating the performance of a machine learning model. Examining all of the models shows that the gradient boosting algorithm outperforms the others.

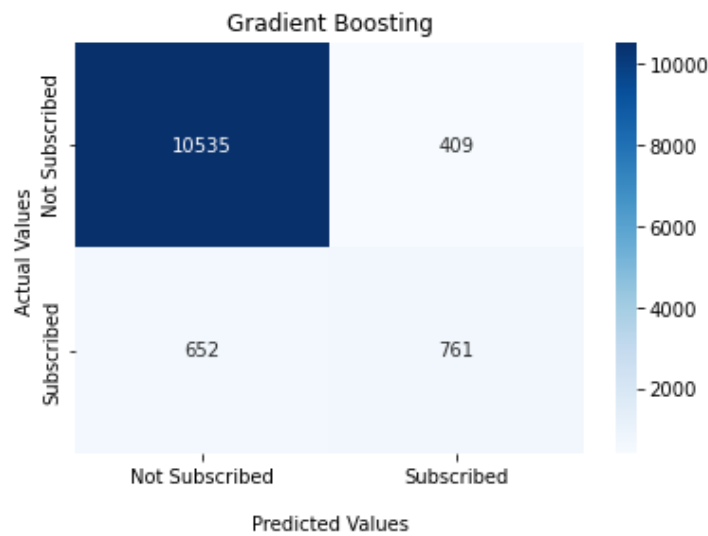
The corresponding confusion matrix below shows a closer look at how the model classified the test set.



Referring to the earlier comment, this model also performs extremely well in reducing the number of false positives while also maintaining a large predictive accuracy (89.67%). Although the duration variable was removed from the analysis it is interesting to compare the results obtained when including the variable in the analysis.



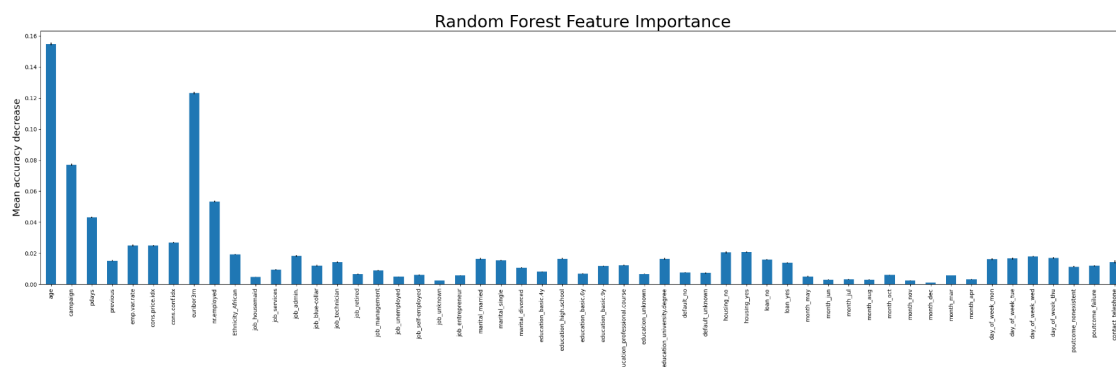
This shows that the algorithms perform significantly better in classification with the gradient boosting algorithm, again, performing the best.



There is an improvement in the overall predictive accuracy (91.39%). Therefore, removing the duration variable from the analysis has led to a more pragmatic model. In both cases, it is interesting to note the model in question 1 has performed quite well by the AUC metric.

Out of the suggested model, the random forest model performs best on accuracy, false positive rate and AUC, so would be our preferred model. However, as we can see above, the XGBoost model performs even better than the random forest model in all aspects bar the false positive rate.

**(c)** The plot of feature importance below shows that age, euribor3m and campaign are the three most important features of the random forest model.





(d) Below is a comparison of the models over all mentioned metrics fit on a 70% training set and tested on a 30% test set, with the green cells being the best model in a given metric.

Model	Accuracy	AUC	TP Rate	FP Rate
Logistic 20%	89.9%	0.78	49.0%	7.8%
Decision Tree	84.2%	0.63	34.7%	9.8%
Bagging	89.0%	0.75	28.4%	3.3%
Random Forest	89.5%	0.78	29.5%	2.7%
XGBoost	91.4%	0.80	54.8%	3.7%

We can clearly see that the logistic regression model performs best out of the suggested models on all metrics other than the false positive rate, meaning we would select this model. However, if we were to extend our potential models to other ensemble methods, we would choose the XGBoost model.

### Question 3

(a) KNN is a supervised machine learning algorithm that can be used to solve both classification and regression problems. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'. Applying the KNN algorithm to the dataset with K equal to 1,3,5 and 10 gives the following confusion matrices:

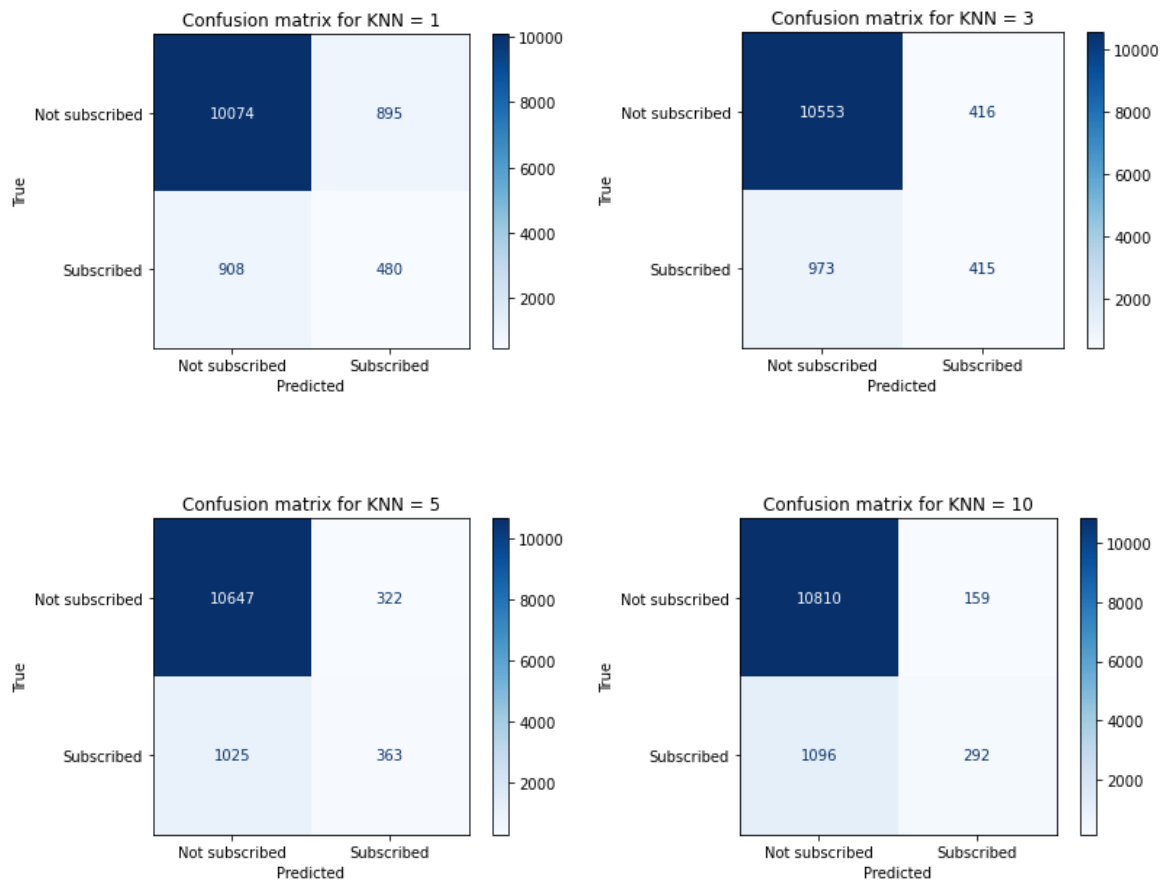


Figure X: Confusion matrices of the KNN algorithm for  $K = 1, 3, 5, 10$

A table of the different metrics of each of the KNN algorithms is presented below:

Model	Accuracy	AUC	TP Rate	FP Rate
KNN = 1	85.4%	0.53	34.1%	23%
KNN = 3	88.8%	0.56	39.0%	23.6%
KNN = 5	89.1%	0.57	41.2%	23.7%
KNN = 10	89.8%	0.60	46.4%	23.9%

As can be seen from the results above, when we increase the value of K from 1 to 10, our predictions become more accurate when predicting the test dataset values. While using a lower value for K gives a more accurate classification score to each of the samples in the training dataset, this is caused by overfitting which leads to an overall lower accuracy score when it comes to the test dataset. A higher value of K produces a more accurately fit model due to majority voting or averaging, giving a more stable and accurate prediction when the model is fit to the test dataset.

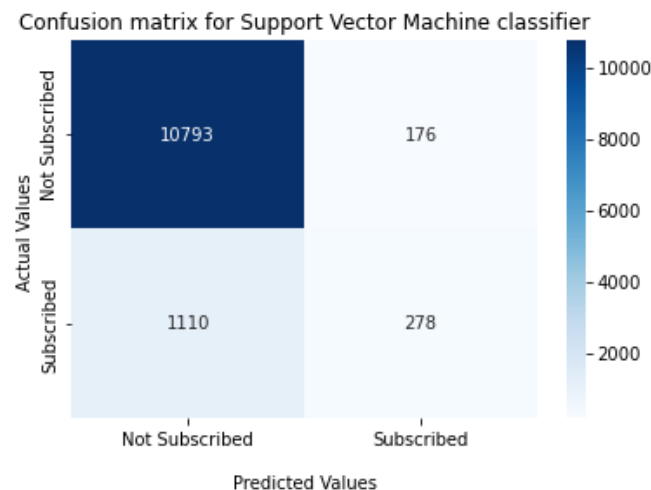
(b) A table comparing the KNN algorithm to the best logistic regression model in question 1 and the best tree-based models in question 2 is displayed below:

Model	Accuracy	AUC	TP Rate	FP Rate
KNN = 1	85.4%	0.53	34.1%	23%
KNN = 3	88.8%	0.56	39.0%	23.6%
KNN = 5	89.1%	0.57	41.2%	23.7%
KNN = 10	89.8%	0.60	46.4%	23.9%
Logistic 20%	89.9%	0.78	49.0%	7.8%
Decision Tree	84.2%	0.63	34.7%	9.8%
Bagging	89.0%	0.75	28.4%	3.3%
Random Forest	89.5%	0.78	29.5%	2.7%
XGBoost	91.4%	0.80	54.8%	3.7%

As can be seen from the table, the KNN models perform marginally worse than the logistic models in Q1 and the tree-based models in Q2. Again, the best performing is the logistic regression model.

#### Question 4

(a) A Support Vector Machine (SVM) classifier was fitted to the dataset. The dataset - excluding the 'Duration' column - was split into a training dataset containing 70% of the data, and a test dataset containing the remaining 30% of the data. Using the 'svm.SVC' function from the 'sklearn' package, a classification prediction was performed on the training dataset producing a prediction set of results. This prediction was then compared to the test dataset created and an accuracy score was calculated.



The SVM classifier model produced a set of results with an accuracy score of 89.6%, similar to other models. Its true positive rate is 20.0% and its false positive rate is 1.6%. Below is a comparison of this model with all others.

Model	Accuracy	AUC	TP Rate	FP Rate
KNN = 10	89.8%	0.60	46.4%	23.9%
Logistic 20%	89.9%	0.78	49.0%	7.8%
Decision Tree	84.2%	0.63	34.7%	9.8%
Bagging	89.0%	0.75	28.4%	3.3%
Random Forest	89.5%	0.78	29.5%	2.7%
SVM	89.6%	N/A	20.0%	1.6%
XGBoost	91.4%	0.80	54.8%	3.7%

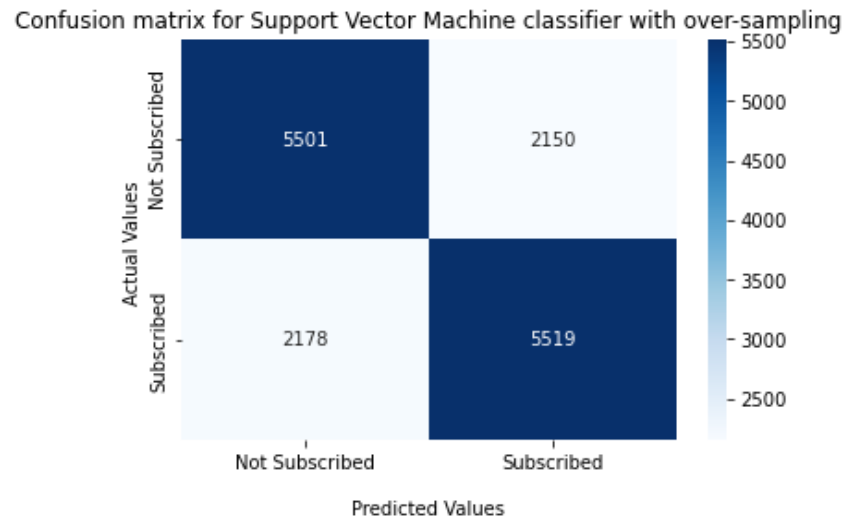
(b) The training dataset provided is highly imbalanced. The dataset as a whole contains just 4,640 positive outcomes out of a total 41,188 observations. The problem with an imbalanced dataset is that most machine learning algorithms operate on the assumption that there are an

equal number of observations in each class. Meaning that a naive application of a model may focus on learning the characteristics of the abundant observations only, neglecting the minority class examples that are of more interest and value. This results in models that have poor predictive performance which is a problem since the minority class is typically more important than the majority class and therefore more sensitive to classification errors. In our case, the desired result is the ability to predict if a client will sign up for a term deposit which only happens ~11.2% of the time in the provided dataset.

This highly imbalanced dataset creates an accuracy paradox wherein an excellent accuracy result may mostly be a reflection of the underlying class distribution. The prediction model that has been trained primarily on one classification outcome, assumes almost all outcomes will fall into that class. For instance, our dataset contains 88.7% non subscription outcomes. Our Support Vector Machine classifier, giving a ~ 89.6% accuracy, may have determined all but ~0.9% of outcomes to be non subscriptions. Meaning that all non subscription outcomes may have been blindly - albeit correctly - predicted, and the remaining observations were predicted with an extremely low accuracy, managing to correctly predict some observations.

One easily implemented solution for an imbalanced dataset is to generate synthetic samples by randomly sampling the characteristics of the observations in the minority class. A systematic algorithm capable of generating synthetic samples is called the Synthetic Minority Over-sampling Technique, or SMOTE. SMOTE is an oversampling method which works by creating synthetic samples from the minority class rather than simply creating copies. The algorithm works by selecting two or more similar samples and altering a sample one attribute as a time by a random amount within the range of the two samples.

By importing the '*SMOTE*' function from the '*imblearn.over\_sampling*' package, the training dataset can be sent through the function to oversample it. Only the training dataset is oversampled as, by its definition, it is used to train the machine learning algorithm while the test set is only used for testing. Once the dataset had been oversampled an SVM classifier model was again fitted to the dataset, producing a result set with a predictive accuracy score of 71.8%. Although this is a lower predictive accuracy score, it may be a better representation of the performance of the algorithm on the dataset as a whole and the true positive rate is much improved, at 71.7%. The false positive rate is increased to 28.1%.



As previously mentioned, an imbalanced dataset creates an accuracy paradox. An over-sampled, or balanced, dataset will allow the machine learning algorithm to learn more accurately how to classify each sample into its correct class. Therefore, it makes sense that the over-sampled dataset has a lower accuracy score, as in fact, it is a better performing algorithm overall and is not falling victim to the accuracy paradox.