Bank teleMarketing success Prediction Project

**Business Understanding**

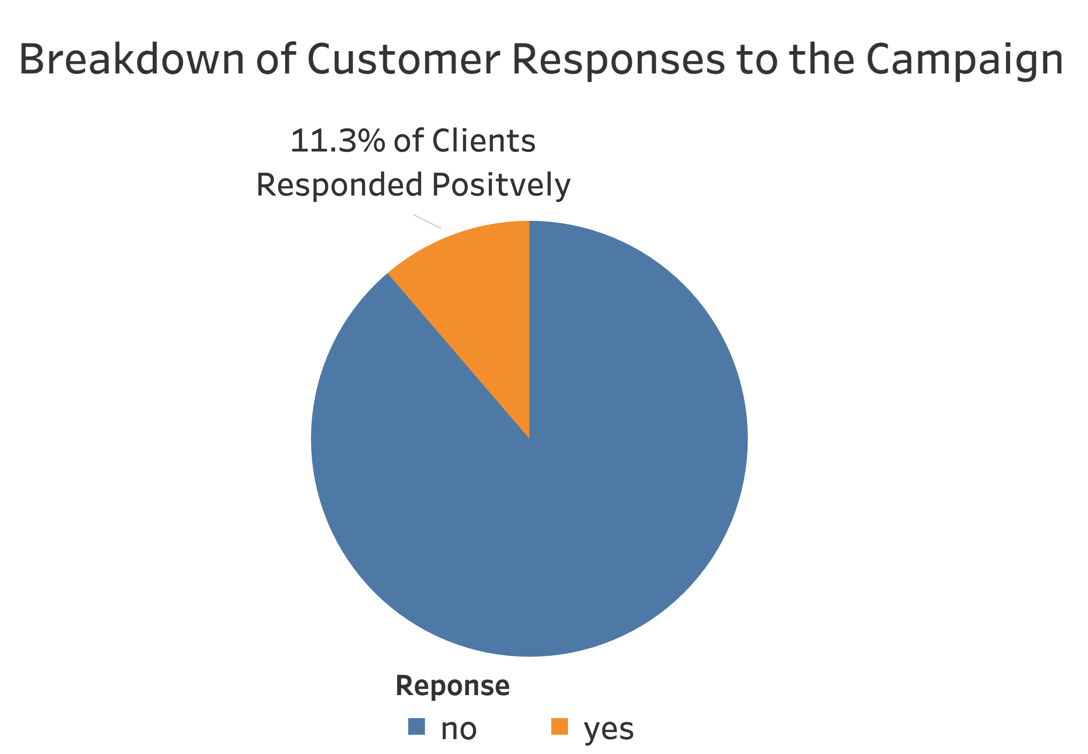
Commercial Banks provide financial services from deposits, checking accounts to personal loans and mortgages to individual consumers and small businesses. They often have direct marketing campaigns to acquire new customers or cross-sell additional products and services to existing customers (dmnnews.com, 2009). These direct marketing efforts have evolved over time, ranging from in-branch communications, direct mails, to emails and social media. With a growing number of communication channels and a saturated marketplace, commercial banks are facing the challenge of declining response rates from these direct marketing efforts (The Financial Brand, 2016).

We would like to use machine learning to help commercial banks achieve better direct marketing results. We want to come up with a model that uses the customer’s demographics information, previous interactions and economical context to predict if the customer will respond positively to the product or services being promoted in the campaign. With this model, commercial banks can have a better understanding of the characteristics that lead to a positive response to the marketing efforts. They can also rank potential customers by the likelihood to respond so that they can determine which customers to target based on marketing budget, leading to a better return on investments for these direct marketing efforts.

**Data Understanding and Exploration**

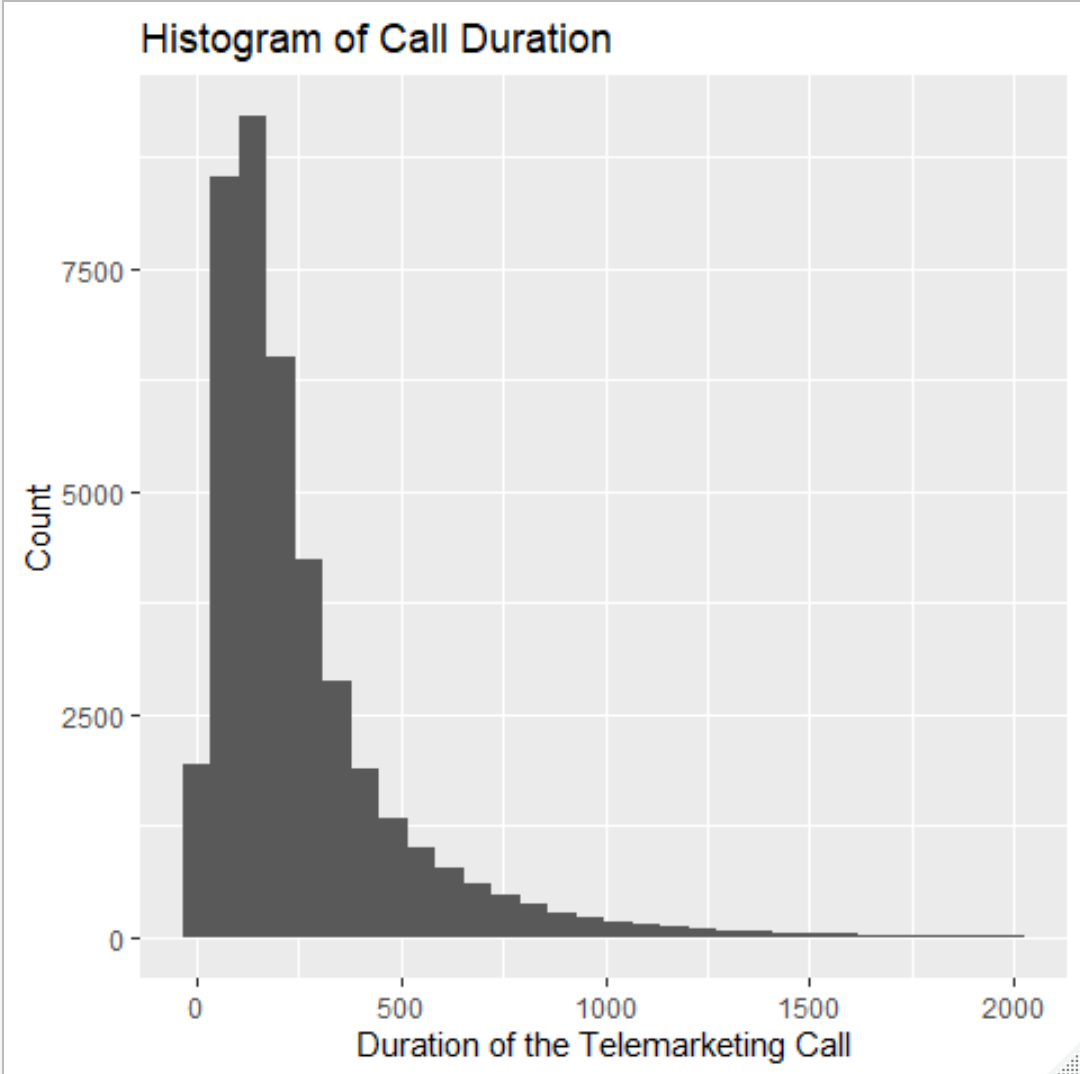
In order to come up with a model to predict customers’ response to direct marketing campaigns, we started with the Bank Marketing data set from UCI Machine Learning Repository (Moro et al., 2014). The data set contains detailed information about telemarketing efforts from a Portuguese banking institution to promote a term deposit product. We have the bank customers’ demographics information, current marketing campaign contact details, previous interactions, social and economic context attributes, and current marketing campaign outcome. With this data set of 41,188 observations and 21 variables, we can use machine learning to explore characteristics that lead to positive response of this telemarketing campaign and build models to predict future success of telemarketing campaigns.

After some initial exploration with the data, we found that the data set is imbalanced with only 11.3% of the 41,188 observations responding positively to the telemarketing campaign, as we can see in the chart below. Depending on the machine learning model, we may need to rebalance the data set to have more positive cases in order to better train the model.



*Figure 1: Breakdown of Customer Responses to the Campaign*

The data set also has information on the duration of the telemarketing call in seconds. As we can see here, it is highly skewed to the right. Most calls were within 10 minutes, however, there are some calls that lasted longer than 30 minutes. This information is highly related to the outcome, but we can’t include the call duration into our models due to data leakage. Data leakage refers to the problem when we include information that we will not otherwise know before we make the decision. In this case, the call duration is not known before we make the call to customers, and we will know whether the customer decides to subscribe to the product or service when the call ends. Therefore, the call duration can be a good reference point, but will not be a predictor in the machine learning models.



*Figure 2: Duration of Telemarketing Calls*

**Data Preparation**

After some explorations with the data, we needed to clean and prepare the data for model building. We looked at each of the categorical values and decided if the number of categories makes sense, because too many categories will lead to overfitting. We also looked at numerical values to make sure they are close to normally distributed. For some of the observations, some categorical values have ‘unknown’. For example, if we look at whether the customer has credit in default, the distribution is as follows. We don’t have information on 20.9% of the customers, so it is too substantial of an amount of data to remove. We then decided to keep ‘unknown’ as a type of category, showing that we don’t have existing information for the customer.

|  |  |
| --- | --- |
| Whether Customer Has Credit in Default | Number of Observations |
| No | 32,588 |
| Unknown | 8,597 |
| Yes | 3 |

*Figure 3: Whether Customer Has Credit in Default Distribution*

For the ‘pdays’ variable that describes how many days have passed since the bank last contacted for a previous campaign, the value ranges from 0 to 27 days with ‘999’ identifying that the customer has never been contacted, and 39,673 of the 41,188 customer (96.3%) of the customers have never been contacted for a previous campaign. We decided that this variable should be a categorical indicator instead of a numerical one, so we converted the values to four categories, as shown below:

|  |  |
| --- | --- |
| Days Since Last Contacted | Number of Observations |
| Less than 5 days | 659 |
| 5 to 10 days | 600 |
| more than 10 days | 256 |
| Never | 39,673 |

*Figure 4: Days Since Last Contacted Distribution*

We also created dummy variables for models that do not work well with categorical values, such as neutral networks.

**Modeling**

To begin our modelling process, we first used a standard scaler in order to fit and transform our X train and test sets. After this we defined an inner and outer cross validation as a KFold where n equals five and the random state is fourty two.

1. **Artificial Neural Network**

We implemented an advanced deep learning Algorithm using keras, which is capable of running on top of TensorFlow. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Our Data preparation for this model included label encoding and one hot encoding for categorical variables and standardization for the train, validation and test datasets (excluding target variable). We customized the layers in this sequential classifier as follows : number of neurons in the input layer is equivalent to number of variables in the train dataset (Having a value of 56). We experimented depth and width for hidden layers and finally figured that 2 hidden layers with 10 neurons in the first and 8 neurons in the second is the most optimal. Anything beyond that would overfit the data while anything below would underfit it. Activation function used is “relu” for hidden layers. Our output layer here has only one neuron since we are looking at a binary classification problem. Acitvation function used for output layer is sigmoid. We then further compiled the neural network with an optimizer : An “adam” ,loss function (which is a “binary cross entropy”) and a f1 score as the metric. We used a batch size as 20 such that weights are updated after 20 inputs from the train data with back propagation. We chose the epochs to be a 100 and noticed that after 25 epochs the result is converges. Finally we applied the model on the test dataset.

1. **Decision Tree**

A popular data science model, decision trees are a supervised learning method used for regressions and classifications. Following a flowchart-like structure, decision trees predict the value of a target variable by learning basic decision rules. We began by using a Grid search Cross Validation technique with a random state of 42 in order to find the highest accuracy. This technique outputted the ‘best’ values for the parameters fed which were: ‘max\_depth’ = 8, ‘max\_leaf\_nodes’ = 20 and ‘mini\_samples\_split’ = 2.

1. **K-Nearest Neighbors (KNN)**

Another algorithm that we implemented was K-NN. Being a non-parametric algorithm, we used it to see if computing distances and obtaining k-nearest data samples would give us a high accuracy value. Similar to the Decision Tree process, we used a GridSearch Cross Validations technique. By defining our X and Y variables, and using n\_splits = 5, we were able to tune the necessary parameters. Based on the parameters we fed the system, it outputted metric = ‘minkowski’, n\_neighbors = 21, weights = ‘distance and a leaf size of 30.

1. **Logistic Regression**

Logistic Regression is another famous statistical model used for classification problems and is based on probability. We began by using a Grid search Cross Validation technique with a random seed of 42 in order to find the highest accuracy. This technique outputted the ‘best’ values for the parameters inputted which were: ‘penalty’ = l2, ‘solver = ‘lbfgs‘ and ‘max\_iter’ = 100.

1. **Naïve Bayes**

A Naive Bayes classifier is a probabilistic machine learning model that is used for classification task. The crux of the classifier is based on the Bayes theorem. We did not apply grid search cv on Naïve Bayes as there weren’t any optimal parameters to be experimented on. Instead we trained the model on train dataset and tested the model on the test dataset in order to see its performance.

1. **Random Forest**

Random Forest is an ensemble tree-based learning algorithm that uses a set of decision trees from a randomly selected subset of a training set. It **aggregates the votes from different decision trees** in order to decide the final class of the test object. We hyper tuned the parameters such as n\_estimators with values such as 50,100,200 ,max\_depth with values such as 1,3,5, 10, 20, 50, min\_impurity decrease with values such as 0.1, 0.01,0.001 and finally max\_features with values such as 'sqrt' and 'log2'. We found that the best hyper parameters are max\_depth =10 ,max\_features=none, min\_impurity\_decrease=0.001, n\_estimators =100

1. **Ensemble Learning**

Apart from the algorithms mentioned above, we also applied ensemble learning models – XGBoost and LightGBM - to further improve the outcomes.

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework and provides a parallel tree boosting that solves many data science problems in a fast and accurate way; LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the advantages of higher efficiency, lower memory usage and better accuracy.

First, we tuned the parameters of XGBoost and LightGBM. For XGBoost, we focused on three important parameters, ‘learning\_rate’, ‘max\_depth’, and ‘min\_child\_weight’, with F1 score to make the parameter selections. Learning rate usually fluctuates from 0.0 to 0.5. Max depth and minimal child weight can range from 1 to 10. We then applied these ranges on cross-validation grid search to find optimal parameters: ‘learning\_rate’ = 0.45, ‘max\_depth’ = 10, and ‘min\_child\_weight’ = 1.

For LightGBM, we focused on three important parameters, ‘learning\_rate’, ‘max\_depth’, and ‘num\_leaves’, with F1 score to make the parameter selections. Learning rate usually fluctuates from 0.0 to 0.5. Max depth can range from 4 to 13. Number of leaves can range from 20 to 170. We then applied these ranges on cross-validation grid search to find optimal parameters: ‘learning\_rate’ = 0.45, ‘max\_depth’ = 11, and ‘num\_leaves’ = 140.

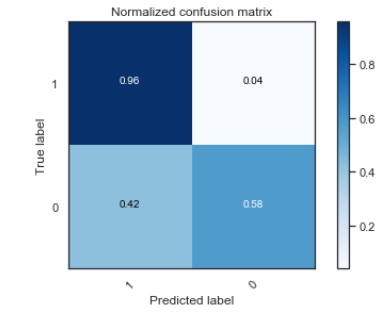
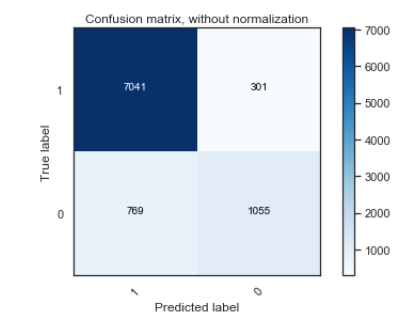
To obtain the possibility of “yes” on the test data and rank them accordingly, we applied LightGBM regressors. We also tuned three parameters: ‘learning\_rate’ = 0.25, ‘max\_depth’ = 13, and ‘num\_leaves’ = 140.

We utilized tuned parameters on the algorithms to train the model using training set and applied it on the test set to evaluate generalization performance.

**Evaluation**

1. **Artificial Neural Network (ANN)**

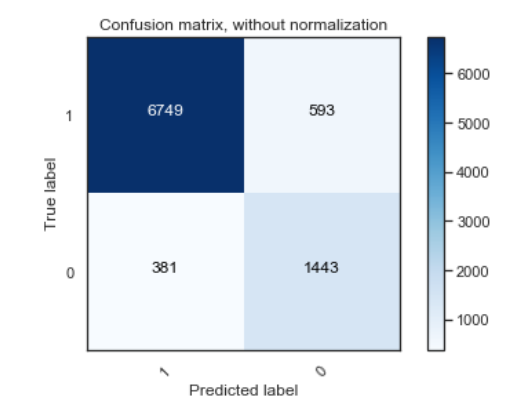
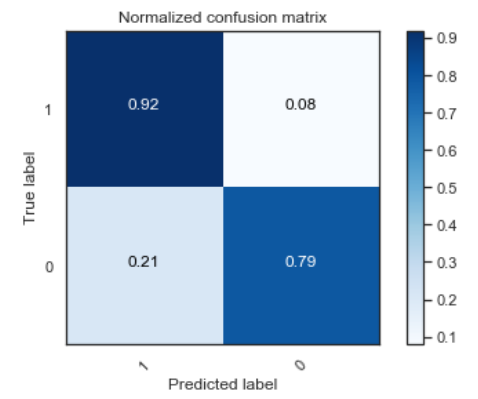
The ANN Classifier outputted an accuracy of 88.3% on the test data and an F1 score of 88%. The false negative rate is 0.04 and the confusion matrix is shown below:

*Figure 5: Confusion Matrices*

1. **Decision Trees**

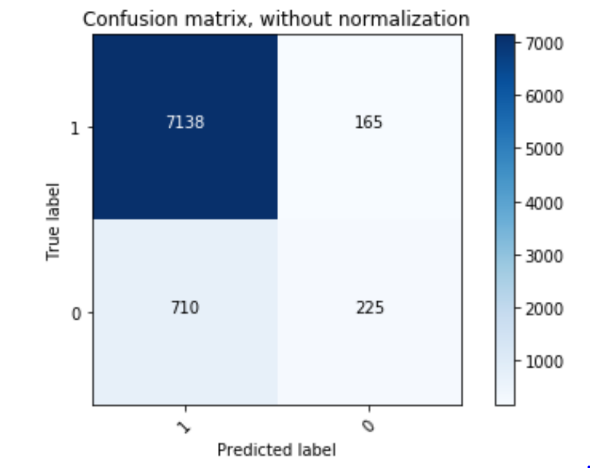
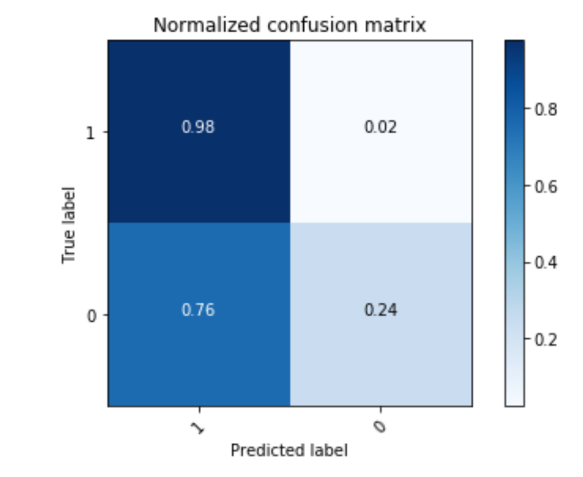
The Decision Tree Classifier outputted an accuracy of 89.4% on the test data and an F1 score of 90%. The false negative rate is 0.08 and the confusion matrix is shown below:

*Figure 6: Confusion Matrices*

1. **KNN**

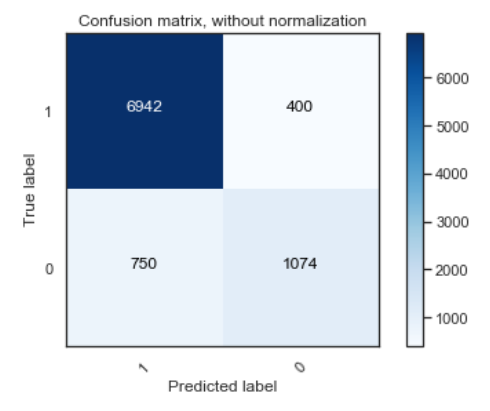
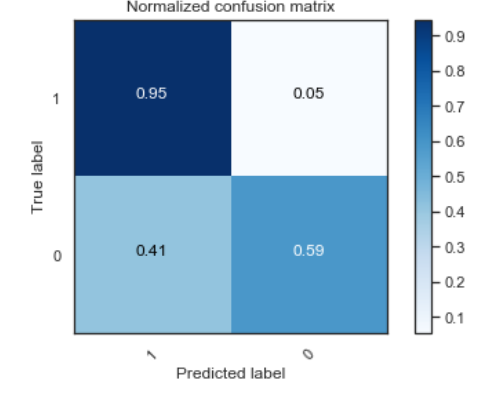
The KNeighbors Classifier outputted an accuracy of 89.4% on the test data and an F1 score of 87%. The false negative rate is 0.02 and the confusion matrix is shown below:

*Figure 7: Confusion Matrices*

1. **Logistic Regression**

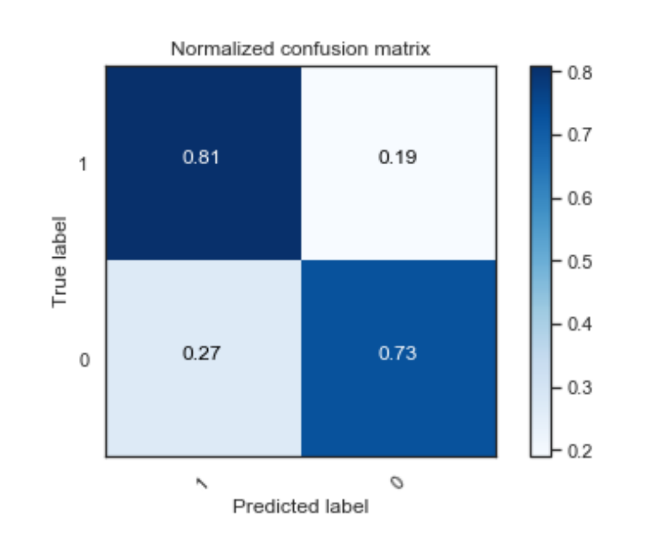
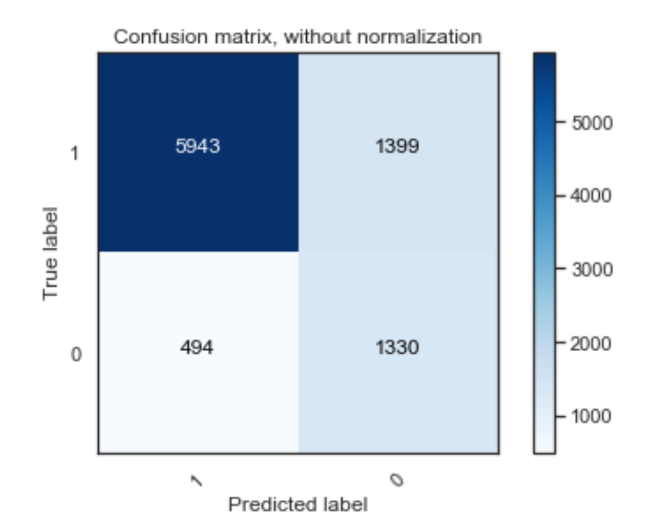
The Logistic Classifier outputted an accuracy of 87.5% on the test data and an F1 score of 87%. The false negative rate is 0.05 and the confusion matrix is shown below:

*Figure 8: Confusion Matrices*

1. **Naïve Bayes**

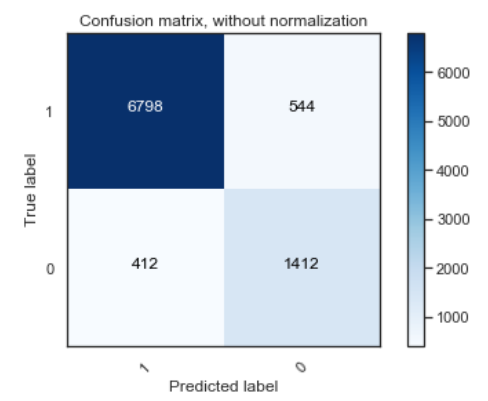
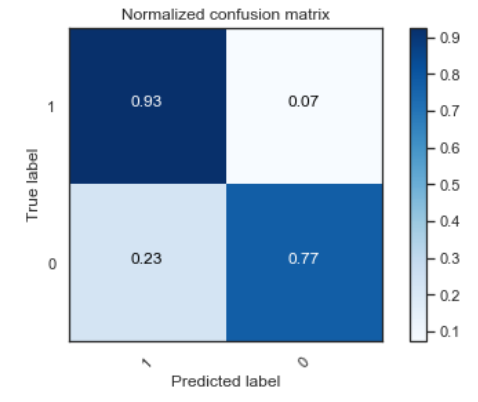
The Gaussian Naïve Bayes Model yielded the lowest accuracy of 79.3% on the test data, an F1 score of 81% and a false negative rate is 0.19. It failed to perform well on the train and test dataset as it failed to surpass the naïve rule which is 80% of accuracy. The accuracy of this model on the train and test dataset are below 80% so we straightaway disregarded this model.



*Figure 9: Confusion Matrices*

1. **Random Forest**

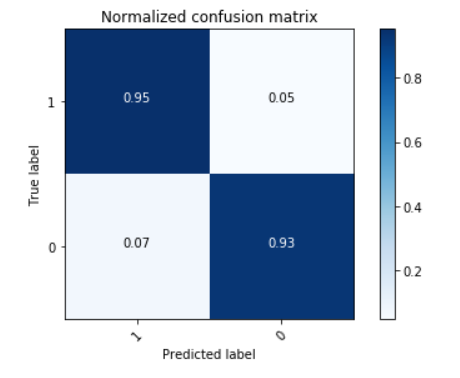
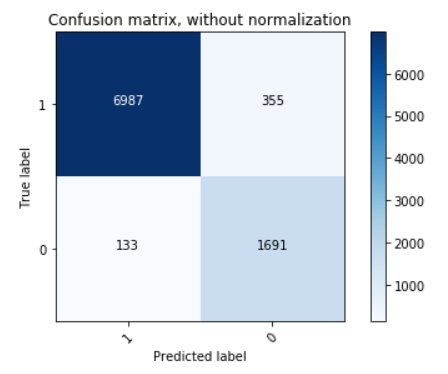
The Random Forest Classifier generated an accuracy of 89.6% on the test data and an F1 score of 90%. The false negative rate is 0.07 and the confusion matrix is shown below:

*Figure 10: Confusion Matrices*

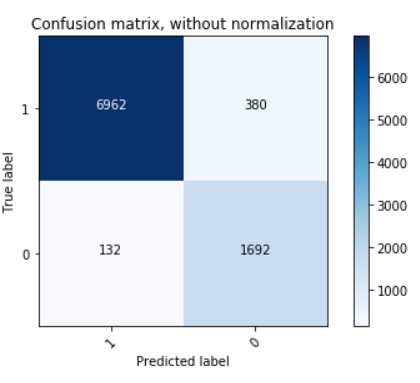
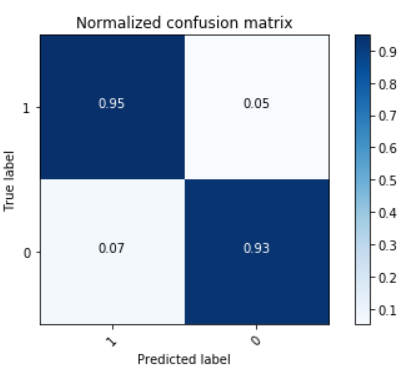
1. **Ensemble Learning**

The XGBoost algorithm yielded an accuracy score of 94.68% on test data, the F1 score is 0.95. These are the confusion matrix with and without normalization and the false negative rate is 0.05:



*Figure 11: Confusion Matrices*

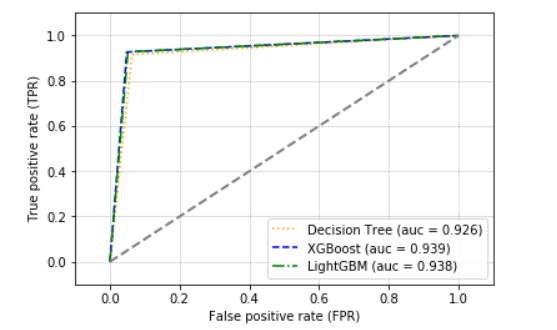
The LightGBM algorithm yielded accuracy score of 94.41 on test data, the F1 score is 0.95. These confusion matrix is listed below with and without normalization and the false negative rate is 0.05:

*Figure 12: Confusion Matrices*

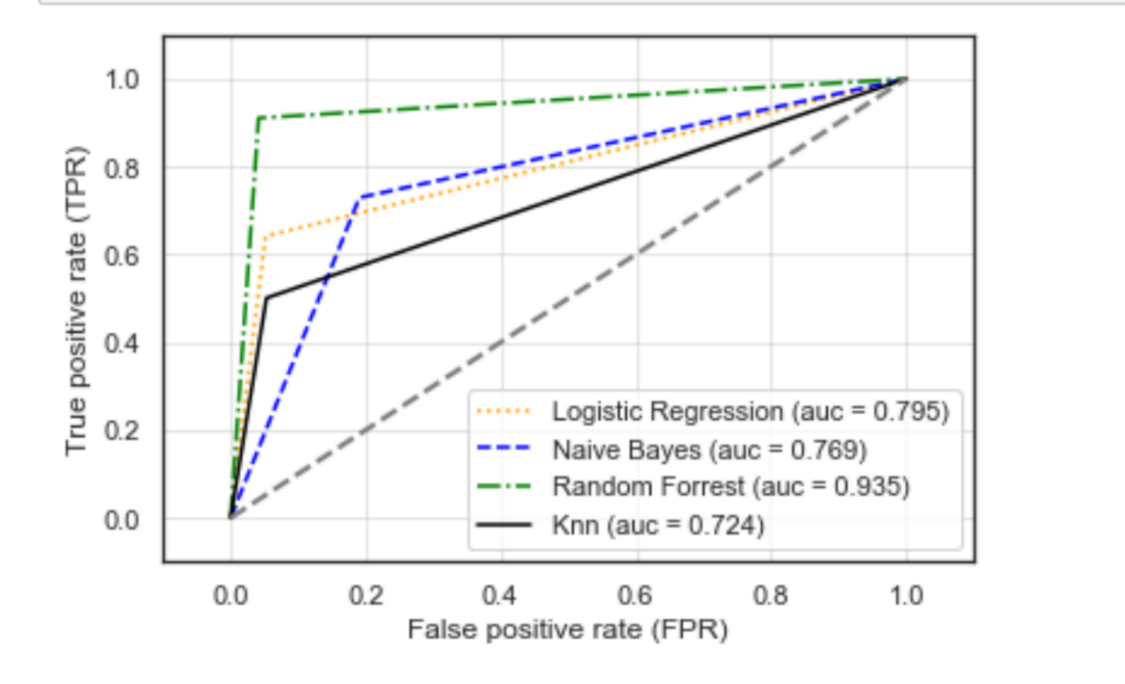
For LightGBM regressor, we calculated the RMSE between predictive outcomes and the actual value, and we get RMSE of 0.222.

To visualize the outcome of our classification performance, we put results into ROC curve and calculated the ROC AUC. We also included a decision tree model as reference. The DT model resulted in 0.926 AUC while XGBoost model resulted 0.939 and LightGBM with 0.938, significant improvement from DT. The ROC curve is shown below:



*Figure 13: ROC Curve*

**ROC AUC for other models:**



*Figure 14: ROC Curve*

**Comparison of Models**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy (Test data) | ROC AUC | FN | FP | Positive Precision |
| Artificial Neural Network | 89.9% | - | 300 | 805 | 90.2% |
| Decision Tree | 89.3% | 0.926 | 593 | 381 | 94.7% |
| Knn | 89.4% | 0.724 | 165 | 710 | 91.0% |
| Logistic Regression | 87.4% | 0.795 | 400 | 750 | 90.2% |
| Naïve bayes | 79.3% | 0.769 | 1399 | 494 | 92.3% |
| Random Forrest | 91.2% | 0.934 | 544 | 412 | 94.3% |
| XgBoost | 94.7% | 0.939 | 355 | 133 | 98.1% |
| LightGBM | 94.4% | 0.938 | 380 | 132 | 98.1% |

*Figure 15: Comparing Models*

**Choosing the best model**

In our business context, we have different costs and benefits. Our cost is huge for falsely predicting that a customer wouldn’t subscribe a term deposit whereas that customer in reality is a potential customer. And our cost is not so huge for falsely predicting that a customer would subscribe a term deposit where as that customer in reality is not a potential customer because sending out promotions is not expensive at all. The medium of promotions is just a phone call. All it costs is some time from an executive but barely any monetary cost. So it’s imperative that our model has minimal false negative rate or higher positive precision. Looking at the above table, we see that boosting algorithms are performing significantly better than any other machine learning algorithms. So we could reject those machine learning algorithms straightaway. For the remaining two boosting algorithms, the performance is fairly similar in terms of every metric. Therefore, we can conclude that LightGBM is the best model, based on optimal performance and computation efficiency.

**Deployment**

Based on the modeling results, we recommend that banks use the LightGBM to predict if customers will respond positively to the term deposit telemarketing campaign. With this model, banks can generate a response rate increase of 22.6% (comparing the test data response rate of 98.1% to the population response rate of 80%), while using 77% of the original marketing expenses. This allows the banks to cut down on marketing expenses when necessary, and at the same time, have a higher response rate and return on investments with these telemarketing campaigns on term deposits. The LightGBM model can also be deployed to generate probability predictions, so that banks can rank customers by probability to respond positively, and target customers with high probability to respond based on their limited marketing budget.

However, the model also has some limitations in deployment. Each telemarketing campaign is slightly different in the product that it promotes and the audience that it appeals to, so it is important to reassess whenever the telemarketing campaign product changes. The communication channel also plays a considerate part in influencing customer decisions, so reassessment is also needed if the communication channel changes from telemarketing to direct mails or etc. With these limitation in deployment scopes, we hope to expand the data set to include data collected from more marketing campaigns and channels, to explore if we can generalize the model to accommodate more marketing types and channels. The data set we used also contains a lot of unknowns. We would like to improve the completeness of the data to have better performance.

**Bibliography**

TODÉ, CHANTAL. “Banks Increase Direct Marketing to Customers.” DMNews.com, 2009, www.dmnews.com/marketing-channels/direct-mail/news/13063167/banks-increase-direct-marketing-to-customers.

Pilcher, Jeffry. “How Banks Can Get Results With Direct Marketing In The Digital Age.” The Financial Brand, 1 Dec. 2016, thefinancialbrand.com/25477/digital-direct-marketing-in-banking/.

“UCI Machine Learning Repository: Bank Marketing Data Set.” UCI Machine Learning Repository: Bank Marketing Data Set, 2014, archive.ics.uci.edu/ml/datasets/Bank+Marketing.

S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014