ITM618 - Marketing For A Banking System - Final Group Project Report

December 7, 2022

Nehal Patel

Aidan Ranjitsingh

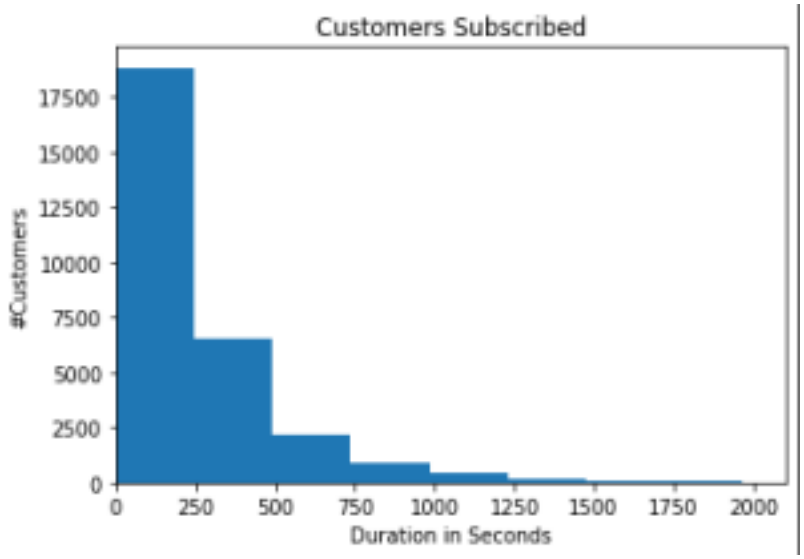
Paul Nguyen

Ali Faraz

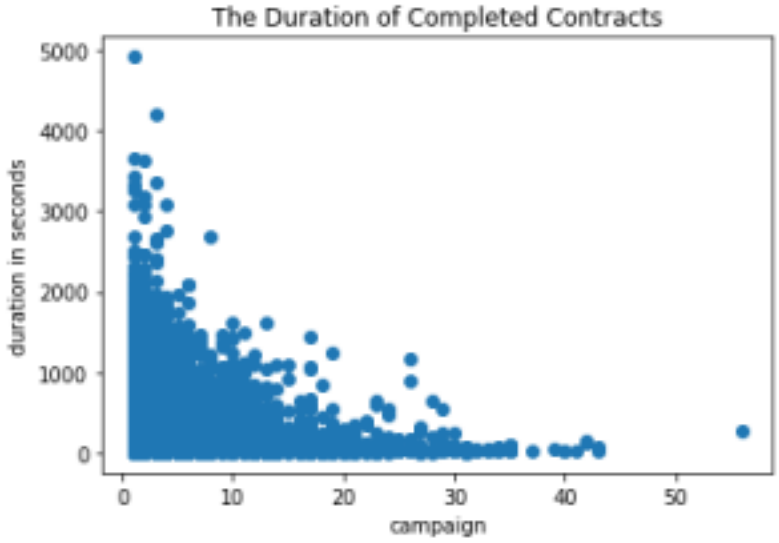
The final group project for ITM618 requested us to predict if a client would subscribe a term deposit or not. The objective of our code is to find that out by testing our algorithm on the target class, subscribed. In our code, the prediction is decided using both the training and test set of data. This report will cover how the data with unknown values was cleaned, how we created our decision tree and KNN classification model and what the result of those methods were.

When going through the dataset from the excel sheet, we were able to find out lots of information, including what types of data we had found, in addition to which types of data had the most variance amongst themselves. Many attributes were object based, meaning that they had qualitative values that varied among each attribute. For example, the Job attribute consisted of over 29000 individuals with varying jobs. This data was converted into an integer that could be used to test the data. Object based data was not the only type of data found, there were also some integer-based data such as age. An example of how we used the data in age was by seeing what the average age was for the individuals who would subscribe to a term deposit and what the standard deviation of ages between them was. In this sheet, the average age was 40.13 and the standard deviation was 9.52. In regards to the data cleaning, we cleaned up the values of the training set, so that we could use it in the test set in order to use for our prediction algorithms.

Additionally in our code, we created a histogram that displayed the number of customers subscribed in relation to the duration in seconds.



With that code we also created a scatterplot graph which further helped to show the campaign in relation to its duration in seconds



The methods we used in predicting if a client would subscribe a term deposit were Decision Tree and KNN. The reason for selecting KNN is that it bases its decision of off a majority, which in this case, the majority we were finding was if most of the target attribute subscribed actually did subscribe or not. KNN is an effective algorithm as it didn't involve us assigning weight to different parameters but is rather based off the actual target attribute.

```
[ ]  y_predicted

     array([1, 1, 1, ..., 1, 1, 1])

[ ]  y_real = testData['Subscribed'].values
     y_real

     array(['yes', 'yes', 'yes', ..., 'no', 'no', 'no'], dtype=object)
```

As seen in this code, the predicted (y_predicted) number of subscribers is all the value 1, which in our code means yes. This means that the algorithm on this run of code predicted that these 6 people would say yes. In contrast, using the actual data (y_real) in these 6 people was that the first 3 subscribed and the last 3 did not subscribe. This is not the full prediction, but rather shows the first and last values of the array to

give you an idea of how the prediction was done. The actual accuracy of the real values to the predicted

values can be found in the code.

The reason we selected decision tree is that it gives a visual representation of how the prediction algorithm works. Decision trees allow you to see the all the attributes in a tree with each attribute having corresponding values.

When this code is run, the tree is generated to give you an idea of how the code comes to its conclusion on if a client will subscribe or not. Internally after this point, we created a confusion matrix which compares the predicted values with the target values (yes or no). This is done to give you an idea of how accurate the prediction algorithm was to the actual values.

With regards to the confusion matrix, there are many values to go over that can compare the test results to the test predictions. When looking at the accuracy between the test results and the predicted results we get a value of 0.9149 meaning that the confusion matrix accuracy is 91.45%. With regards to the error rate, it would be the opposite of the accuracy, being that it is 8.5%. These results show that our prediction model has an accuracy of 91.45% with an error rate of 8.5% indicating that the model is very accurate. There are two more values to look at when looking at the confusion matrix, being precision and recall. Precision shows what percentage of positive classifiers labelled by the prediction algorithm were actually positive. This value in our case is 65.89% meaning that 65.89% of the values that were labelled as positive were actually positive. When looking at recall, we get a value of 64.1% meaning that 64.1% of selected positive values were classified as positive. The precision and recall being close together means that the prediction algorithm is accurate.

When comparing the two different classification methods it is evident that the KNN algorithm does the same amount of work with less processing power. It does fundamentally the same predictions as

the decision tree and outputs its values in an array allowing quick comparisons between both the predicted array and the actual array. The part that is lost in the KNN classification method is the visualization. With the decision tree, you can see exactly how a prediction is made by following the tree and seeing what values are used to come up with that answer. The other part which makes decision trees less efficient is also using a confusion matrix with it. The confusion matrix compares the actual values with the predicted values, and because of having to build both the tree and then actually comparing the results, much more processing power is required over KNN.

In conclusion, both methods proved to come up with similar prediction accuracies, therefore meaning that they were both equally as effective at coming up with the result. The decision tree had positives being that it came up with a visual aspect to that prediction data, that allowed you to follow along better and see how the algorithm came up with its result at the cost of more computing power and requiring more formatting. KNN was effective as it came up with the result and put it into an easy-to-read array at the cost of having to scroll through thousands of rows of data if you want to compare each individual spot in the array. In the case of KNN, if you aren't needing to compare each individual part of an array, you can just calculate the accuracy, which will give you the probability that a client will subscribe to the term. In the future, in order to improve on the models, one way would be to reduce redundancy in the data, being that there were many unknown values that needed to be accounted for. Having this data taken out would make for a more accurate end result as it would then be fully based off of the actual data.