*Andrew Farrell*

*S1511335 | afarre203@caledonian.ac.uk*

*Data Analytics*

Coursework 2.1 – Practical Data Analytics

“I declare that all work submitted for this coursework is the work of Andrew Farrell alone except where explicitly stated otherwise.”

Table of Contents

[Chapter 1 Description of the Problem 2](#_Toc27529332)

[Chapter 2 Construction and Tuning of SVM classifier 4](#_Toc27529333)

[2.1 Data Pre-processing 4](#_Toc27529334)

[2.1.1 Outliers 4](#_Toc27529335)

[2.1.2 Encoding Categorical Values 5](#_Toc27529336)

[2.1.3 Separation of Input and Output 6](#_Toc27529337)

[2.2 Implementation and Tuning of SVM 7](#_Toc27529338)

[2.2.1 Tuning the Model 7](#_Toc27529339)

[Chapter 3 Testing Results 9](#_Toc27529340)

[Chapter 4 Comparison and Discussion 11](#_Toc27529341)

[Chapter 5 References 12](#_Toc27529342)

[Chapter 6 Appendices 13](#_Toc27529343)

[Appendix A 13](#_Toc27529344)

[Appendix B 18](#_Toc27529345)

[Appendix C 20](#_Toc27529346)

[Appendix D 21](#_Toc27529347)

[Figure 1 - Missing Values in the Dataset 2](#_Toc27529366)

[Figure 2 - Variation of Categorical Values 3](#_Toc27529367)

[Figure 3 - Function to Calculate Outliers 4](#_Toc27529368)

[Figure 4 - Boxplot for Outliers in Age 4](#_Toc27529369)

[Figure 5 - Boxplot for Cons.conf.idx Outliers 5](#_Toc27529370)

[Figure 6 - Encoding Ordinal Categories 5](#_Toc27529371)

[Figure 7 - Encoding Nominal Categories 6](#_Toc27529372)

[Figure 8 - Separation of Input Variables and Output Target 6](#_Toc27529373)

[Figure 9 - Implementation of SVM 7](#_Toc27529374)

[Figure 10 - 60/40 Train/Test Split Results 9](#_Toc27529375)

[Figure 11 - 70/30 Train Test Split Results 9](#_Toc27529376)

[Figure 12 - 80/20 Train/Test Split Results 10](#_Toc27529377)

[Figure 13 - Random Forest Classifier Output 11](#_Toc27529378)

[Table 1 - Attribute Information 2](#_Toc27529379)

[Table 2 - Comparison of Confusion Matrix 11](#_Toc27529380)

# Description of the Problem

The dataset provided from [Moro et al., 2014] and is concerned with a marketing campaign of a Portuguese banking institution, where data was captured on phone calls to customers offering them a term deposit. Moro et al. state that often more than one phone call was required to get in touch with the bank’s customers. The details of the dataset overall are listed in table 1 below.

Table 1 - Attribute Information

|  |  |  |
| --- | --- | --- |
| **Banking Dataset** | | |
|  | **Count** | |
| Number of Attributes | 21 - (11 numeric, 10 categorical) | |
| Number of Instances | 41,188 | |
| **Output Values** | | |
| **Output Variable** | **Count** | **Overall %** |
| Client Subscribed - Yes | 4,640 | 11.27 |
| Client Subscribed - No | 36,548 | 88.73 |

The dataset contains 41,188 records with the aim of the dataset determining whether or not a customer subscribed to a term deposit after being contacted through the banks marketing campaign. For this project the dataset will be used with a Classification algorithm that can be used to predict the outcome of whether or not a client would subscribe to a term deposit, through analysing the attributes identified in appendix A. One of the main considerations identified in table 1 is the lack of diversity in the outcome variable, as the dataset is vastly dominated by an outcome of ‘no’ (88.73%) compared to ‘yes’ (11.27%). On first exploration of the dataset it appeared to not have any ‘NULL’ or ‘NaN’ values, however the dataset has ‘unknown’ values in the attributes; job, marital, education, default, housing, and loan. The number of ‘unknown’ values are represented in figure 1 below. The total number of ‘unknown’ values is equal to 12,718 with the largest amount of ‘unknown’ values being found in the ‘default’ attribute.

A screenshot of a cell phone

Description automatically generated

Figure 1 - Missing Values in the Dataset

Figure 2 displays the variation that each of the categorical values have, with most of the categorical values considered to be nominal values with the exception of ‘education’ which would instead be considered ordinal as there is different educational levels (University Education > School Education etc.). Using a Support Vector Machine (SVM) algorithm to classify the data means the categorical values will have to encoded during data pre-processing in order to be interpreted accurately by the algorithm.

A screenshot of a cell phone

Description automatically generated

Figure 2 - Variation of Categorical Values

The distribution of the numerical values is displayed in Appendix A which contains histograms displaying the variation. From the histograms contained in Appendix A it is clear that for the majority of attributes there is no clear ‘*normal’* distribution, with the exception of the Age attribute which appears to follow the bell-shaped curve. The main outcome from these histograms is highlighted in the ‘pday’ attribute which contains information about how long ago the customer last contacted the bank, however there is a majority of entries at the value 999. The reason for this is that 999 is the default value for customers who haven’t contacted the bank previously.

# Construction and Tuning of SVM classifier

This section contains the information on how the SVM classifier was implemented and tuned in order to perform classification on the dataset, as well as the relevant data pre-processing that took place to prepare the dataset for the for the algorithm.

## 2.1 Data Pre-processing

The first step of the implementation process involved processing the attributes in the dataset into a relevant format to be used in the SVM classifier. During this stage there was three main features which involved dealing with outliers, encoding categorical values, before finally separating the dataset into input attributes and the output target.

### 2.1.1 Outliers

The first task of the data pre-processing phase involved calculating outliers in the numerical attributes. This was achieved through implementing an outlier tukey function that is able to identify outliers through the use of quartiles and interquartile ranges as shown in the implemented code in figure 3.

A screenshot of a cell phone

Description automatically generated

Figure 3 - Function to Calculate Outliers

**Age**

The first attribute that returned with outliers in the data was age, which returned with any data subject with an age greater than or equal to **70** was considered to be an outlier. A boxplot was also used to verify that this age range would be considered as outliers, which was confirmed. This returned a number of *469* instances out of the *41,188* that started in the dataset therefore is equal to *0.011%.* Due to this the age outliers were removed from the dataset, which reduced the number of instances in the dataset to ***40,719***.

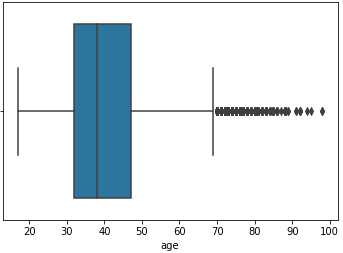


Figure 4 - Boxplot for Outliers in Age

**Cons.conf.idx**

The second attribute that returned with outliers was cons.conf.idx which returned with any data subject that has a consumer confidence index greater than **-30** to be considered an outlier. Again, a boxplot was used to verify this which is displayed in figure 5 below and again confirmed that this was indeed correct. A number of *627* out of *40,719* instances were returned as being outliers in the cons.conf.idx attribute, which makes up *0.015%* of instances. Due to this having a minimal percentage of impact over the number of instances in the dataset, these outliers were again removed which left **40,092** instances in the dataset.

A screenshot of a social media post

Description automatically generated

Figure 5 - Boxplot for Cons.conf.idx Outliers

### 2.1.2 Encoding Categorical Values

Using an SVM classifier meant that categorical string values cannot not be interpreted and would have to processed into an understandable format for the algorithm. To do this, two types of encoding were used. Firstly, ordinal values were encoded using the find and replace method (Moffitt, 2019) which was achieved through the code highlighted in figure 6. Using this method involved matching the categorical string to its number representative (March is the 3rd month therefore attached to 3 etc.). Only two attributes were encoded using this method which were ‘month’ and ‘day\_of\_week’ as these were the only ordinal categories in the dataset.

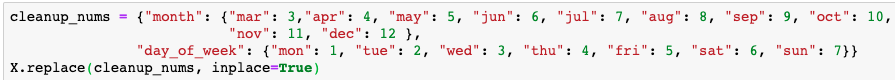


Figure 6 - Encoding Ordinal Categories

The second method used for nominal categories was One Hot Encoding (Moffitt, 2019), which involves creating a new attribute for each of the unique values in each category and then assign these with binary values (0 = False, 1 = True). This was implemented through the function identified in figure 7 and was used on the remaining categorical attributes identified in Appendix B.

A screenshot of a cell phone

Description automatically generated

Figure 7 - Encoding Nominal Categories

### 2.1.3 Separation of Input and Output

The final task of the data preparation stage was to separate the target output variable ‘y’ from the input variables. This was achieved through the code highlighted in figure 8, which also identifies how the output variable was normalised to include binary values rather than the string counterpart.

A screenshot of a cell phone

Description automatically generated

Figure 8 - Separation of Input Variables and Output Target

## 2.2 Implementation and Tuning of SVM

A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 9 - Implementation of SVM

First of all, the implementation involves splitting each dataset into a training and testing dataset. The ratio used for this was 80% for training and 20% for testing as shown in figure 9. This means that 32,074 instances would be used to train the SVM classifier to help it identify which input attributes determine the output target. This then leaves 8,018 instances for the SVM to be tested on to determine if the classifier can successfully predict the outcome target when only the input attributes are known.

To make sure the tuning of the SVM model the same ‘random\_state’ was used meaning each time that the classifier function ran it would take the same sample of data, therefore preventing any differences purely from the changes in training data.

### 2.2.1 Tuning the Model

Initially the model was implemented using the ‘rbf’ SVC kernel with a ‘C’ value of 1. When this ran on the 80/20 split of data an accuracy of 89.72% was achieved. From this the model was then tuned to improve accuracy.

1. The SVC kernel was changed to ‘linear’ with the ‘C’ level remaining at 1.

This increased the accuracy of the SVM algorithm by 0.7% to achieve an accuracy of 90.42%

1. The SVC kernel then remained the same with an increase of the ‘C’ value to 10.

This increased the accuracy of the SVM algorithm to 91.65%.

1. The ‘C’ parameter was then increased to 100 while the kernel remained the same.

This reduced the level of accuracy in the SVM classifier to 91.58%, due to overfitting to the training data that comes with the increase in the ‘C’ parameter (Medium, 2019). This led to the ‘C’ parameter being reduced back to 10 as it provided the most accurate prediction.

# Testing Results

To test the results of the SVM model there was three methods used; Cross Validation, Confusion Matrix and a Classification matrix. These were tested across a variation of training and test splits to determine the most accurate version of the model to use. An explanation of the methods used to validate the SVM classifier is contained in Appendix C

A close up of text on a white background

Description automatically generated

Figure 10 - 60/40 Train/Test Split Results

Figure 10 above highlights that the 60/40 train/test split had the following:

* Has a cross validation mean of 90.17%
* Confusion Matrix with predictions of 14,504 correct outputs and 1,533 incorrect outputs
* Achieved precision score of 91%, recall of 90% and an F1-score of 90%

A screenshot of a cell phone

Description automatically generated

Figure 11 - 70/30 Train Test Split Results

Figure 11 above highlights that the 70/30 train/test split had the following success:

* Has a cross validation mean of 90.63%
* Confusion Matrix with predictions of 10,962 correct outputs and 1,066 incorrect outputs
* Achieved precision score of 90%, recall of 91% and an F1-score of 89%

A screenshot of text

Description automatically generated

Figure 12 - 80/20 Train/Test Split Results

Figure 12 above highlights that the 80/20 train/test split had the following success:

* Has a cross validation mean of 90.75%
* Confusion Matrix with predictions of 7,350 correct outputs and 669 incorrect outputs
* Achieved precision score of 91%, recall of 92% and an F1-score of 90%

From the tests completed as highlighted in figure 10, 11 and 12 it is evident that the best performing model was the 80/20 training/test split which had the highest success for each of the categories tested as well as scoring the overall highest accuracy out of the three tested.

# Comparison and Discussion

A screenshot of a cell phone

Description automatically generated

Figure 13 - Random Forest Classifier Output

The second model that was implemented is the Random Forest Classifier available in Appendix D, which makes predictions based on a number of decision trees. Comparing the accuracy of both shows that the second model has a minimal improved accuracy (0.8%) over the SVM model. The Classification Report also provides higher values for the second model as highlighted in table 2 below. The comparison of confusion matrix in figure 12 and 13 also highlight that the second model had a higher success rate.

Table 2 - Comparison of Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Report Output** | **Model 1** | **Model 2** |
| Precision | 91% | 92% |
| Recall | 92% | 92% |
| F1-score | 90% | 92% |

Overall from the test conducted which include the overall accuracy, the confusion matrix and classification report the model which provides the most successful and therefore the optimum choice of model would be model 2 – the Random Forest Classifier.

# References

GeeksforGeeks. (2019). *Confusion Matrix in Machine Learning - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ [Accessed 18 Dec. 2019].

Medium. (2019). *In Depth: Parameter tuning for SVC*. [online] Available at: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769 [Accessed 18 Dec. 2019].

Moffitt, C. (2019). *Guide to Encoding Categorical Values in Python - Practical Business Python*. [online] Pbpython.com. Available at: https://pbpython.com/categorical-encoding.html [Accessed 18 Dec. 2019].

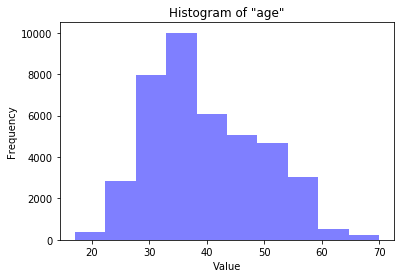
Scikit-yb.org. (2019). *Classification Report — Yellowbrick v1.0.1 documentation*. [online] Available at: https://www.scikit-yb.org/en/latest/api/classifier/classification\_report.html [Accessed 18 Dec. 2019].

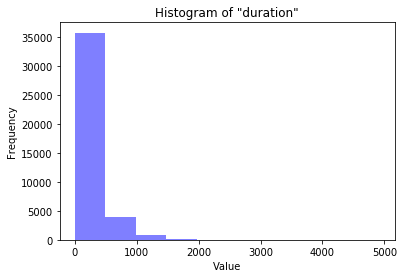
# Appendices

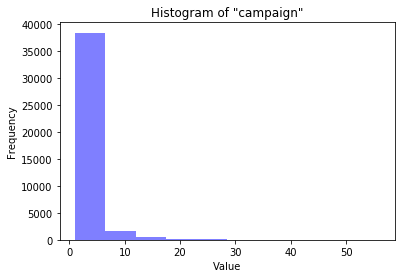
The python notebook used for the implementation of this project is available at:

<https://github.com/andy-farrell/DA_2.1>

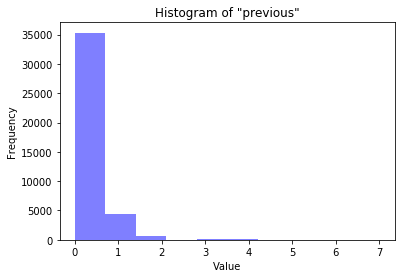
## Appendix A

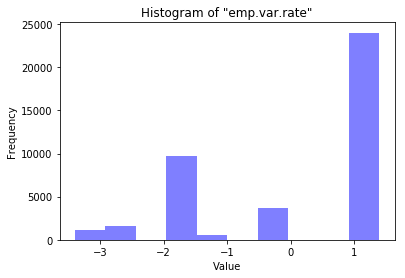


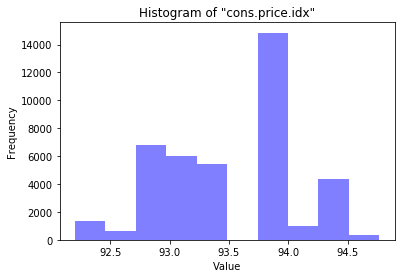


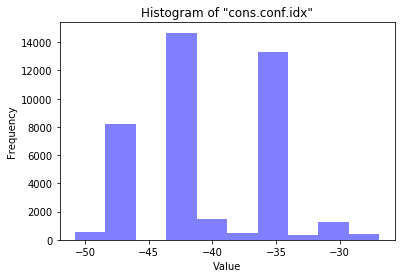


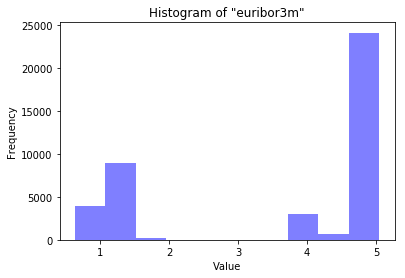


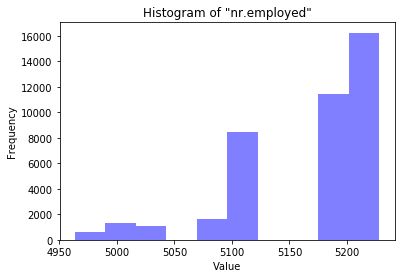








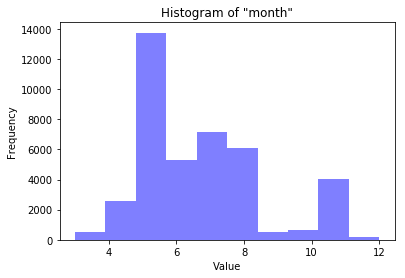


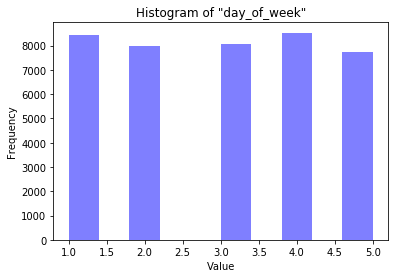


## Appendix B

A screenshot of a cell phone

Description automatically generated





## Appendix C

**Confusion Matrix:**

A screenshot of a cell phone

Description automatically generated

A confusion matrix provides a summary of predictions from the classification problem. The table above shows how the predictions that are made are stored where it outlines the predictions that the classifier model made and whether or not they were actually correct or not (GeeksforGeeks, 2019).

**Classification Report/Matrix:**

This report outlines the precision, recall, F1 score and support score for the model that has been implemented. The precision refers to the classifiers ability to not label an instance positive when it is actually negative. This effectively means that for all instances classified as positive what percentage where actually correct. Recall is the classifiers ability to find all the true instances. This means that for all instances that were classified as positive, what percentage was correctly classified. F1 score provides a ‘weighted harmonic mean of precision’ which is ranked between 0.0 and 1.0. The support feature of the classification report shows the number of occurrences for each of the output targets and can help identify imbalance in the dataset (Scikit-yb.org, 2019).

## Appendix D

A screenshot of a cell phone

Description automatically generated

The Random Forest classifier was implemented using a train and test split the same as the SVM classifier at 80/20. Again the same random\_state was selected to make sure the results were based on a similar selection of the data set to provide an accurate comparison as possible.