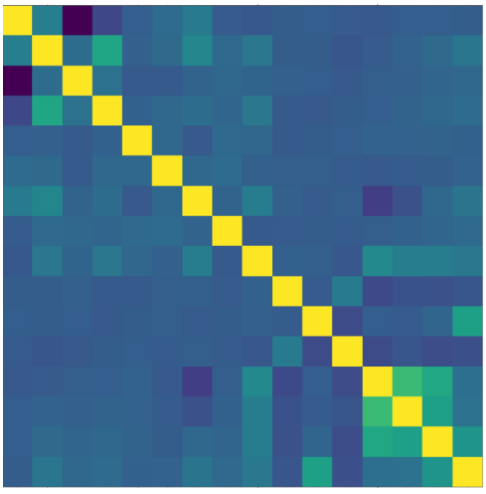

Correlation Matrix of the entire dataset

FBA: Coursework - 1

COURSEWORK 2019

TAHSINUR RAHMAN KHAN

Contents

1 Introduction

In this report we analyzed the historical data of 5000 marketing calls to customers for selling the "N/LAB" platinum deposit" financial product. Our task was to build a suitable model based on the historical data so that it could be used to predict the outcome of future calls made to customers from varying socio-economic backgrounds. There were in total of 15 different input variables pertaining to different information about the customers, as well as the records of previous marketing attempts made to them.

The report is split up into 6 parts, namely: (A) Summarization of the dataset; (B) Initial exploration of the dataset; (C) Model Evaluation; (D) Final Assessment; (E) Model Implementation; and (F) Business Case Recommendation.

2 Section A: Summarization

In this section we performed preliminary statistical analysis of the entire dataset. Starting with checking the balance of the data and the statistical analysis of the features present in the data.

The first step we performed was to check the balance of the data. Simple counting of the classes present in the output column revealed that the data was highly unbalance as shown in figure 1. The number of marketing calls that were unsuccessful were much higher than the number of marketing calls that were successful, 4414 to 586 respectively.

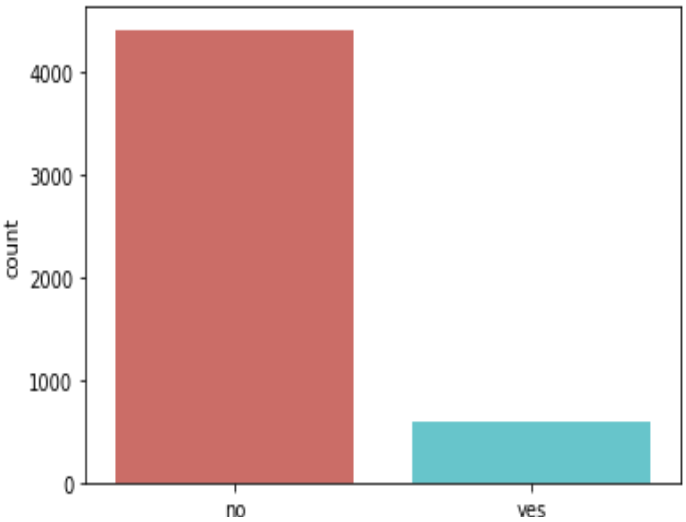

Figure 1 Number of successful calls (yes) - 586 to number of unsuccessful calls (no) – 4414

2.1 Finding the Highest Correlating Features

Apart from being an imbalanced, the dataset contained a mixture of nominally ordered, categorical text features and numeric, ordinal features. Since we were going to applying machine learning modelling to the dataset for final prediction, we needed to process the data from a heterogeneous mixture to a more homogeneous one. For this purpose, we gathered all the categorical, text features and applied additive smoothing technique [1] to encode the data. This was done instead of other popular encoding techniques such, as one-hot encoding or label encoding, in order avoid problems that might appear later on in the workflow such as curse of high dimensionality and improper encoding [2].

```
Top Absolute Correlations
pdays      previous     0.550655
           poutcome     0.427876
job        education    0.421423
age        marital      0.412446
duration   y            0.388829
previous   poutcome     0.385763
poutcome   y            0.317107
contact    pdays        0.254289
job        housing      0.234074
age        job          0.190604
```

Figure 2 Highest Correlating Features

After the initial pre-processing, the correlation matrix was computed and the top 10 highest correlating features were found. It can be seen from figure 2, that the *pdays* feature produced the highest (Pearson's) correlation with two other features - *previous* and *poutcome.* Moreover, the output feature *y* was highly

correlated to the *duration* feature. Upon close inspection it was noticed that the *duration* feature listed the duration of the last marketing call with each customer. Since this cannot be used for future predictions, this column was dropped. The remaining features were then put through further statistical testing.

2.2 Feature Analysis and Weight Factors

Following our additive smoothing on the categorical data, we wanted to check whether our data had been adequately weighted or not. According to additive smoothing, the features that had the higher ratio of *yes:no* would have the higher weights. Table 1 shows a granular level analysis of one of the categorical features – *Education*. Each feature was broken down into their corresponding categories and relative weights were applied to each of those categories. Final weight factor of each feature contained the combination of each category present in the feature. The weight factor for the *education* feature was calculated out to be 0.14909.

EDUCATION:		
Values	Fraction	Weight
primary	0.0893	0.097363
secondary	0.1039	0.105264
tertiary	0.1556	0.149109
unknown	0.107	0.112932

Table 1 Granular level analysis of Education feature

Figure 3 shows the categorical features with the highest fraction of *yes'es.* The blue bar represents the size of the fraction and the orange bar represents the corresponding weights that had been applied. It was seen that the heights of the blue and orange bars closely correlated with each other, which meant proper weighing and encoding methods were followed.

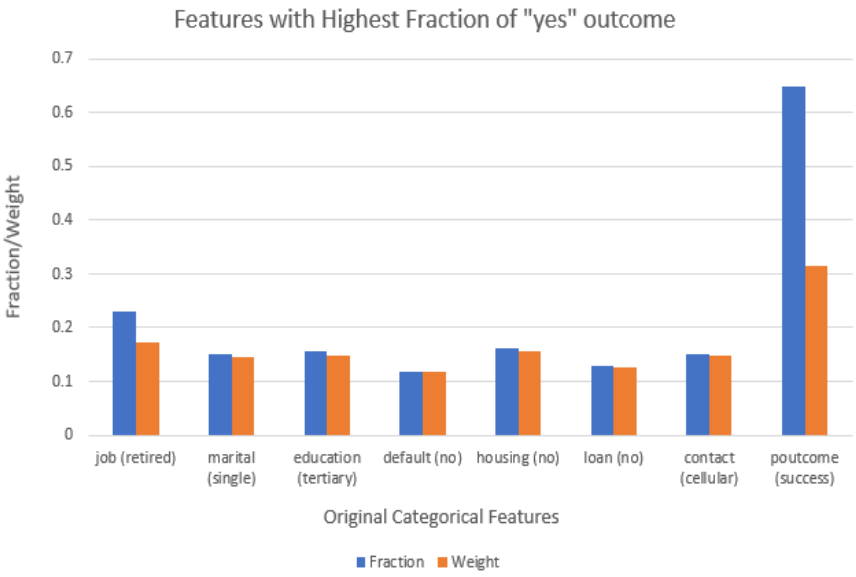

Figure 3 Comparison between the category importance to a successful outcome and their relative weights

Further statistical analysis were performed to identify the outlier in each of the numeric features to identify any outliers and the mean values of their corresponding outcomes. Figure 4 shows one such box plot. From the figure it could be seen that the average age of customers that resulted in successful sale was 37 yo and the average age for unsuccessful call was 39.

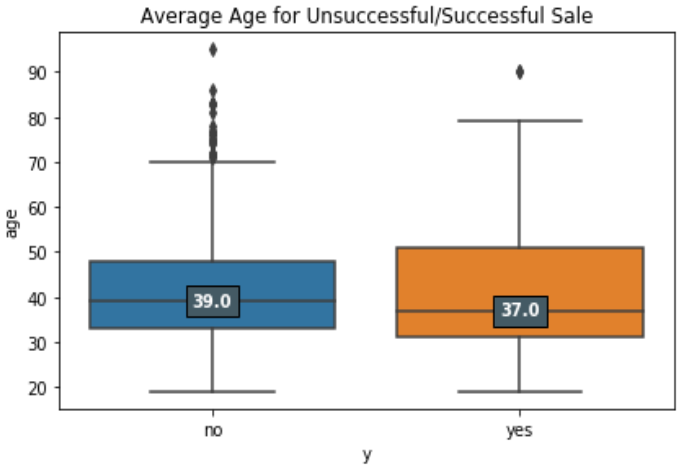

Figure 4 Average Age for unsuccessful and successful sale

Closer inspection of the numeric features revealed that the average value for both yes-no outcomes for *pdays* feature stood at -1. A value of -1 in this feature meant that a given customer had not been previously contacted. A numeric value on this column represented *the number of days that had passed by after the client was last contacted in a previous campaign*. This meant that the column consisted of both continuous values (count of the real days passed) and categorical values of -1. It was suspected that high occurrence of -1's in the *pdays* feature pulled the average values of both yes-no outcome towards it. Figure 5 shows the boxplot of *pdays* column and corresponding spread. It was noticed that high number of statistical outliers were reported to the high number of -1's in the column. The corresponding outcome of no:yes for *pdays = -1* stood at 3680 : 374. Therefore, only 946 instances contained real valued days.

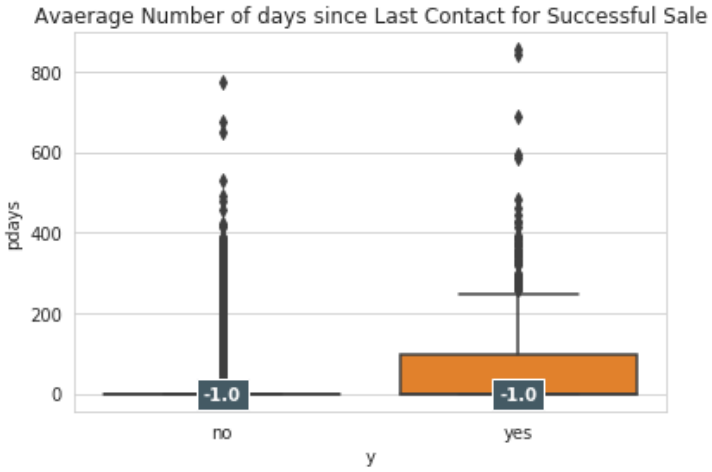

Figure 5 Boxplot for "pdays" feature

Due to characteristic of the values, described above, in the *pdays* and *duration* feature; we created a copy of the entire dataset without it.

3 Section B: Exploration

3.1 Exploratory Accuracy Analysis

For this part we continued our preliminary analysis of the dataset using a decision tree. As mentioned in the previous section, we created two datasets each of which had their categorical features properly scaled using additive smoothing. The first dataset (df_post) contained all the features except for *duration* and the second dataset was a copy of the first one, minus the *pdays* feature (df_post_wo_pdays). The two datasets were then modelled using a decision tree and a justification was made on which one to keep for further analysis.

The df_post dataset was again further split into two additional datasets – one that contained all instances of customer who had not been contacted before (df_post1) and the other dataset consisted of customers

who had been contacted before (df_post2). In df_post1 dataset, the *pdays* feature was removed since all the *pdays* values were equal to -1. Table 2 shows the accuracies obtained for each of the datasets. Table 2 shows the corresponding accuracies for each of the datasets when modelled with a decision tree.

Decision Tree Accuracies	
Dataset	Accuracy
df_post	0.822666667
df_post_wo_pdays	0.818666667
df_post1	0.832374692
df_post2	0.746478873

Table 2 Accuracies of each dataset modelled with a Decision Tree

3.2 Preliminary Modelling with Decision Tree

As can be seen from the analysis, the df_post1 represented the highest accuracy. However this dataset excluded all the customers who had been previously contacted. Therefore, we sacrificed the slight gain in preliminary accuracy and decided to take the df_post dataset for further analysis.

After choosing a suitable dataset for further analysis, we performed another modelling using a decision tree, but this time the maximum depth was set to 3, to analyze the features. Figure 6 shows a decision tree, split according to the gini index which means a node's impurity, for the top 4 layers.

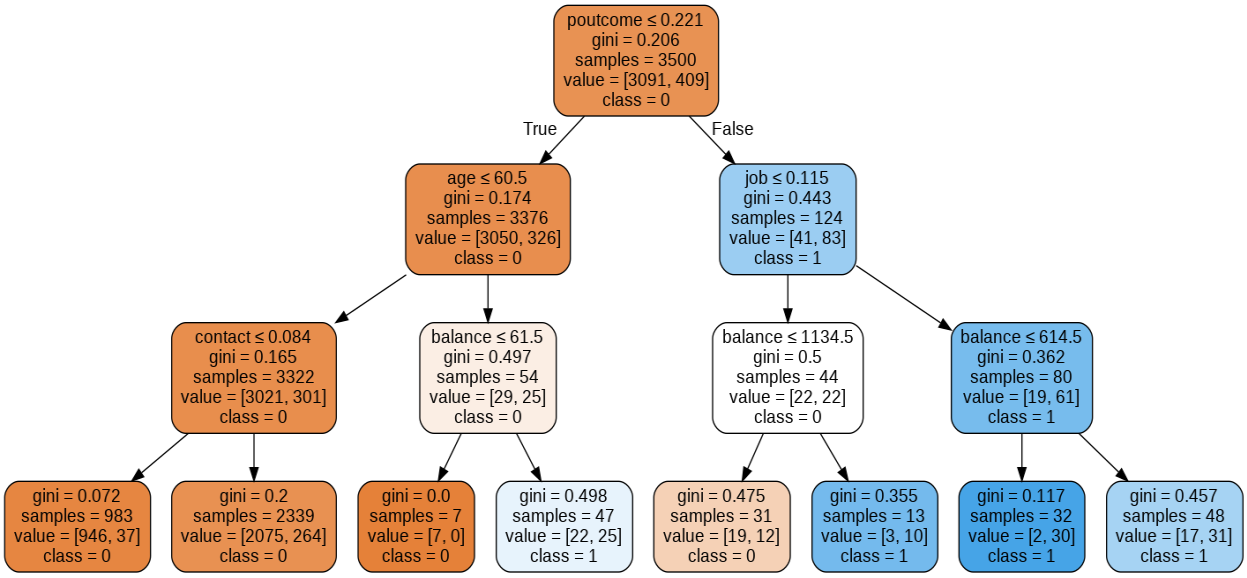

Figure 6 Decision Tree with only the first 4 layers

Each layer in a decision tree is split according to the maximum gain in information or maximum decrease in impurity. That means irrelevant attributes that are poorly associated with target labels will produce poor splits and would not result in maximum decrease in impurity [3]. Hence, the decision trees tend to place relevant features at the top [4]. Therefore, according to the decision tree diagram above, the *poutcome, age, job, contact,* and *balance* features seem to be the most relevant. Therefore a new dataset – df_post4 was created with only those "relevant" features.

In the next section, we consider the two datasets – df_post and df_post4 for analysis.

4 Section C: Model Evaluation

For our analysis we have chosen to analyze our datasets using three classification techniques: Logistic Regression; Naïve Bayes Classifier; and k-Nearest Neighbours. In this section we also performed data oversampling in the training set using the ADASYN module since our dataset was highly imbalanced.

4.1 Data Balancing using ADASYN:

During our initial analysis using logistic regression, we found that the df_post dataset produced 88.13% accuracy. However the confusion matrix produced told a different story. Table 3 shows the confusion matrix of the this preliminary modelling. As can be seen from this confusion matrix, the true outcome of y = 1 was never correctly predicted! This showed that the dataset was very imbalanced and needed rebalancing before proper modelling could be performed.

		PREDICTED	
		0	1
TRUE	0	1322	1
	1	177	0

Table 3 Confusion Matrix produced by using logistic regression on for df_post

Since the number of y = 1 class was vastly under-represented in our dataset, so we performed synthetic oversampling using ADASYN algorithm. ADASYN works by finding the n-nearest neighbors in the minority class and draws a line between the neighbors. Then the algorithm generates synthetic data samples along the line, in a random order – which is meant to correctly represent the dataset in high dimensionality dataset [5]. The ADASYN is an alternative to the SMOTE algorithm.

To properly configure the use of ADASYN on our dataset, we first split the dataset into train, validation and test sets in 60:20:20 ratios. Then the ADASYN algorithm was applied only on the training set in order to stop "*bleeding*" of synthetically produced data into the validation and test sets [6]. This was done to ensure that our model could be generalized properly on unseen data.

4.2 Logistic Regression

The validation and test set accuracies of applying logistic regression on df_post dataset were 66.5% and 66.8%. The recall rates for validation and test sets were 46.5% and 50.45% We also noticed from the confusion matrix, that our data was not more evenly balanced among the *yes* and *no* classes. Table 4 shows the confusion matrix after applying logistic regression on the df_post dataset after oversampling on the training set using ADASYN.

		PREDICTED	
		0	1
TRUE	0	613	278
	1	54	55

Table 4 Confusion matrix using logistic regression on df_post dataset, post ADASYN over-sampling

The ADASYN oversampling technique was applied to df_post4 dataset as well. Figure 7 and 8 shows the precision, recall, accuracy and f-1 scores of applying logistic regression on df_post and df_post4 datasets.

	precision	recall	f1-score	support
0	0.92	0.69	0.79	891
1	0.17	0.50	0.25	109
accuracy			0.67	1000
macro avg	0.54	0.60	0.52	1000
weighted avg	0.84	0.67	0.73	1000

Figure 7 Statistics after application of logistic regression on df_post

	precision	recall	f1-score	support
0	0.92	0.84	0.87	891
1	0.22	0.37	0.27	109
accuracy			0.79	1000
macro avg	0.57	0.60	0.57	1000
weighted avg	0.84	0.79	0.81	1000

Figure 8 Statistics after application of logistic regression on df_post 4

4.3 Naïve Bayes:

Figure 9 and 10 shows the statistics of applying Naïve Bayes on df_post and df_post4 datasets.

	precision	recall	f1-score	support
0	0.91	0.27	0.42	891
1	0.11	0.77	0.20	109
accuracy			0.33	1000
macro avg	0.51	0.52	0.31	1000
weighted avg	0.82	0.33	0.39	1000

Figure 9 Application of Naive Bayes on df_post

	precision	recall	f1-score	support
0	0.92	0.23	0.37	891
1	0.12	0.83	0.20	109
accuracy			0.30	1000
macro avg	0.52	0.53	0.29	1000
weighted avg	0.83	0.30	0.35	1000

Figure 10 Application of Naive Bayes on df_post4 dataset

4.4 K-Nearest Neighbors:

Before doing the kNN analysis, we had to make sure that were choosing the suitable number of neighbors as a hyper-parameter for the kNN model. This step was crucial for the analysis. For that we measured the error rates versus the number of neighbors (K). Figures 11 and 12 show that for the df_post dataset, the error rate stabilized between 0.4 to 0.5 for K > 150 and for df_post4 dataset, the error stabilized to be in between 0.45 to 0.55 for k > 300. Therefore k = 151 and k = 301 were chosen for kNN modelling on df_post and df_post4 datasets respectively. It is generally a good idea to choose odd number for even number of classes [7]. Figure 13 and 14 show the statistics on applying kNN on df_post and df_post4 datasets.

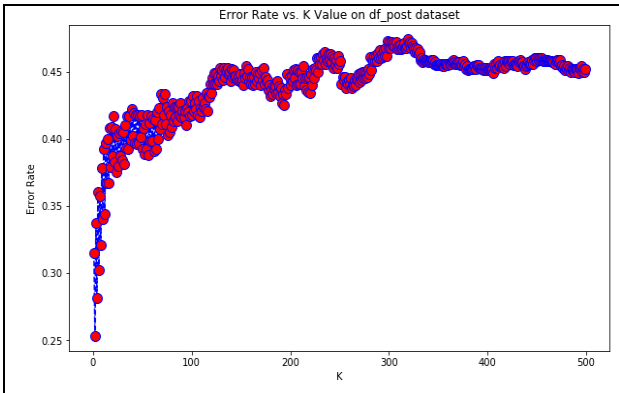

Figure 11 K value stabilized for k > 150 on df_post

Figure 12K value stabilized for k > 300 on df_post4

	precision	recall	f1-score	support
0	0.91	0.54	0.68	891
1	0.13	0.56	0.21	109
accuracy			0.54	1000
macro avg	0.52	0.55	0.45	1000
weighted avg	0.82	0.54	0.63	1000

Figure 13 kNN on df_post

	precision	recall	f1-score	support
0	0.92	0.50	0.65	891
1	0.13	0.63	0.22	109
accuracy			0.52	1000
macro avg	0.53	0.57	0.44	1000
weighted avg	0.83	0.52	0.60	1000

Figure 14 kNN on df_post4

By considering the above statistics, it was determined that logistic regression on the df_post dataset gave the most reliable accuracy, precision, f1 and recall rates. Hence we choose this mode and dataset combination l for analysis of our final model.

5 Section D: Final Assessment (Final Model Selection, Grid Search and Normalizing the dataset)

Before developing our final model, we normalized our df_post dataset. Only the features that contained numeric values were normalized. Since the categorical features were already normalized when additive smoothing was applied on them so they were left unchanged. ADASYN was applied again on the normalized data. This normalization was done since we were going to be using logistic regression model – which is a linear model, for our final analysis. Linear models require that the dataset be normalized so that all the values can be made unit-agnostic and homogeneous for proper modelling [8].

We tuned our final model, on the normalized df_post dataset, using grid search. The hyper-parameters that were explored were the type of solves, the regularizer and the C value which defines the search space of the hyper-parameters. Table 5 shows the hyper-parameters that corresponded to the best logistic regression model.

Hyper-param Search	
Solver	liblinear
Regulari:	l2
C	127.4274986

Table 5 Optimal hyper-parameters for the best logistic regression model

Using these hyper-parameters we got validation and test set accuracies of 63% and 64.1% repestively. The recall rates were 60.5% and 69.7% on validation and test sets. Although the accuracy scores in our final logistic regression model were a bit diminished, compared to our initial logistic regression model (as shown in figures 7 and 8), the recall rates were much higher! Figure 15 shows the statistics of applying

	precision	recall	f1-score	support
0	0.94	0.63	0.76	891
1	0.19	0.70	0.30	109
accuracy			0.64	1000
macro avg	0.57	0.67	0.53	1000
weighted avg	0.86	0.64	0.71	1000

Figure 15 Statistics of the final logistic regression model

our final linear regression model on the df_post dataset. The corresponding confusion matrix is shown in table 6.

		PREDICTED	
		0	1
TRUE	0	565	326
	1	33	76

Table 6 Confusion matrix of the final model

Figure 16 shows the ROC curve of our final model. The AUC value of our final model came at 0.67. The blue trend line corresponds to our final model, while the red dotted line corresponds to that of a random classifier's performance on the same dataset. Since our curve is placed much higher than that of the random classifier, so we can guarantee that our model will perform better.

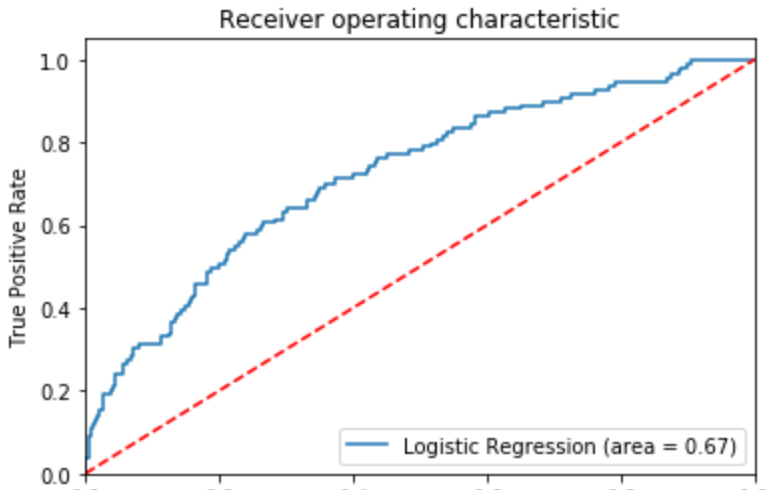

Figure 16 ROC curve of final model

6 Section E: Model Implementation

The final model was then tested against the whole dataset using a 10 fold cross validation. The final statistical measures of the dataset are shown in figure 17.

We can see that our final model will generalize well for new dataset. We expect our dataset to show similar performance measures on any new dataset of the same modality.

	precision	recall	f1-score	support
0	0.61	0.64	0.62	4414
1	0.63	0.61	0.62	4506
accuracy			0.62	8920
macro avg	0.62	0.62	0.62	8920
weighted avg	0.62	0.62	0.62	8920

Figure 17 Performance measures of our final model after training on entire dataset using 10 fold cross validation

7 Section F: Business Case Recommendation

The performance of our final model could be improved further by training it on a bigger and better balanced dataset. For future analysis we have few recommendations for exploration such as: dimensionality reduction techniques; Bayesian based smoothing on categorical data; Splitting the dataset between old customer (those who have been contacted before) and new customers and then analyzing the different sets separately; Further exploring the sampling hyper-parameter space of ADASYN and SMOTE; and perform Recursive Feature Elimination for identifying the most optimal feature space.

8 References:

[1] https://www.wikiwand.com/en/Additive_smoothing

[2] https://www.kdnuggets.com/2017/04/must-know-curse-dimensionality.html

[3] Introduction to Data Mining (Second Edition), Tan et al., chapter 3, pp - 144

[4] http://www.learnbymarketing.com/603/variable-importance/

[5] https://medium.com/coinmonks/smote-and-adasyn-handling-imbalanced-data-set-34f5223e167

[6] https://beckernick.github.io/oversampling-modeling/

[7] https://discuss.analyticsvidhya.com/t/why-to-use-odd-value-of-k-in-knn-algorithm/2704

[8] https://stats.stackexchange.com/questions/189652/is-it-a-good-practice-to-always-scale-normalize-data-for-machine-learning