Comparing The Performance of Multilayer Perceptrons and Support Vector Machines in Predicting The Success of Marketing Campaigns in an Individual Sale.

Abstract

In this paper, we aim to investigate the performance between two supervised algorithms, in predicting the success of a telemarketing campaign based on several features of the clientele. The two algorithms compared here are the Feedforward Multilayer Perceptron (MLP) and Support Vector Machines (SVM). Different hyperparameter configurations were explored using grid search and validated using a 5-fold cross validation. The tested results were then compared using Balanced Accuracy Scores and Confusion Matrices. The results indicated that MLP was the best model for our problem domain, but test results were underwhelming showing the need for future work.

1. Introduction

Marketing selling campaigns are one of the key strategies used to grow business' revenues and is one of the largest costs incurred by them depending on the industry. Direct marketing is often used when a business have a product which they are trying to push. However, these campaigns generally have a low individual success rate and utilise the advantage of large numbers to hit the outreach quota making it extremely inefficient and cost ineffective. Therefore, if a predictive model can be made, the number of clienteles that needs to be contacted would be reduced significantly.

The goal of this paper is to critically evaluate the use of two algorithms, Multilayer Perceptron (MLP) and Support Vector Machines (SVM) in predicting if a client has subscribed to a term deposit given a number of features regarding that client. Various hyperparameters were considered to find the optimal models for both algorithms and their performance was investigated using the metrics accuracy score and confusion matrix.

1.1 Multilayer perceptron (MLP)

Multilayer Perceptron's are a commonly used supervised artificial neural network used for pattern recognition. It is composed of 3 types of layers, an input layer used to take in datapoints, one or more hidden layers that extracts the necessary information during learning and assigns a modifiable weight coefficient that is determined during training, and an output layer which returns the predicted values [1].

Initially, during the training process, the values from the inputs are modified by the weight coefficients which are randomly assigned by the model to determine the output value - this is known as the forward-pass. Output values are then compared to the true outputs to determine the degree of error which is sent back through the network. During this process, the weights are adjusted accordingly depending on the optimizing function employed and is known as back-propagation. Once training is complete and the weight coefficients are determined, the output layers will then return values associated to each class.

1.2 Support Vector Machines (SVM)

SVM is a classification model that transforms the input values into high-dimensional feature spaces, using nonlinear mapping the way this is conducted depends on the kernel employed. It finds the

decision boundary which best separates the data classes making it applicable for binary classification problems. Utilising SVM's provide two key advantages, when working with high-dimensional data, it only requires a few data points i.e., those which are required for the support vector, secondly with the use of the dot product SVMs can theoretically extrapolate to any number of hyper-dimensions making it computational efficient [2].

2. Dataset

Column Name	Categorical Values	Transformed Values
jobs	['admin.','bluecollar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown']	[-1,……,12]
martial	['divorced','married','single','unknown']	[-1,…,2]
education	['primary', 'secondary', 'unknown', 'tertiary']	[-1,…,2]
default	['no', 'yes']	[0,1]
housing	['no', 'yes']	[0,1]
loan	['no', 'yes']	[0,1]
contact	['cellular', 'unknown', 'telephone']	[-1,…,1]
poutcome	['no', 'yes']	[0,1]

Table 1. Transformed Categorical Features before scaling was performed.

The dataset used was obtained from the UCI Machine Learning Repository. It contains 41188 datapoints with 17 features for each observation. 10 of the features were categorical with the remaining features being numeric. The features 'day' and 'month' indicating when the participants were last contacted were removed as the features 'pdays' provided the number of days since the clients were last contacted by the bank making these two features redundant. Before modelling could be conducted, the remaining categorical features were first transformed into numeric values (**Table 1**), all features were then scaled between the values of 0 and 1. The dataset was then split into training (60%), validation (20%) and testing (20%) sets. Initial explorations of the dataset revealed heavy class imbalances ("no":88.3%, 'yes':11.7%), which would lead to biases in our models' predictions. Therefore, to circumvent this, SMOTE was applied to our training dataset which resulted in an increase of 20808 datapoints and a 50/50 split between the two classes.

3. Methods

The pipeline for our model comparison consists of two major parts- model selection and algorithm selection. Model selection was performed using a grid search to determine the best hyperparameters for both models. Training and validation were conducted using a 5-fold stratified cross validation which divided the training data into 5 samples each containing the same number of classes which was executed natively by sklearn's "gridsearchCV" function. This approach is generally preferred over k-fold cross validation as it often yields higher levels of accuracy and sensitivity. The average validation accuracy was then used to determine the best hyperparameters of each model [3].

Once the optimal hyperparameters were determined, the models were then retrained on the same training set and were both tested with the same test dataset. To evaluate the performance of our algorithms, confusion matrices accuracy score was used as our scoring methods of choice.

3.1 Architecture and Parameters used for MLP

For our modelling purposes an MLP with 2 hidden layers utilising the Adam optimization algorithm was used. A sigmoid activation function was used as it only outputs values between the ranges of 0 and 1 making it suitable for binary classification problems.

To avoid the possibility of our models overfitting during training early stopping was employed. During training, 10% of the training data was allocated as a validation set which was used to score the models performance during each epoch. The training session would then then terminate if the validation score did not improve after 10 consecutive runs, preventing our models from overfitting to the dataset.

Grid search was conducted on the hyperparameters: learning rate, size of the hidden layers, and the beta coefficients of the Adam optimizer, with the hopes of optimising for the models learning speed, generalization and predictive capabilities.

3.2 Parameters used for SVM

Unlike MLPs, the architecture of SVMs do not need to be predefined. The main parameters which must be defined are the kernel functions and box constant. Determining the best kernel function to use would require iterative testing on the dataset as its performance would depend on the complexity and distribution of the data itself. The box constant is a cost on misclassification errors and aids in the model's regularization. As the value of the constant increases, separation becomes stricter with less support vectors being assigned, the drawback being the increase in computation times.[4]

4. Results, Findings and Evaluation

4.1 Model Selection

	MLP				
β1	β2	Hidden Layer Size	No. Epochs	Learning Rate	Accuracy Score
0.7	0.79	(15,15)	400	0.015	0.8426
0.7	0.89	(15,15)	500	0.015	0.8406
0.8	0.99	(15,15)	400	0.015	0.8406
0.7	0.99	(15,15)	400	0.015	0.8404
0.8	0.79	(15,15)	400	0.015	0.8403
0.9	0.79	(10,10)	500	0.015	0.8402
0.9	0.89	(15,15)	300	0.015	0.8402
0.9	0.79	(15,15)	300	0.015	0.8401
0.9	0.89	(15,15)	400	0.015	0.8400
0.9	0.99	(15,15)	300	0.015	0.8394

SVM			
Box Constant	Kernel Function	Polynomial Order	Accuracy Score
0.020	linear	1	0.7972
0.020	poly	3	0.7953
0.015	poly	3	0.7863
0.015	linear	1	0.7859
0.020	poly	2	0.7849
0.010	poly	3	0.7738
0.015	poly	2	0.7714
0.010	linear	1	0.7688
0.020	poly	1	0.768514
0.010	poly	2	0.755497

Table 2. Best hyperparameters for each model

With reference to **Table 2**, we see that the best scoring MLP had a hidden layer size of 15 neurons, a learning rate of 0.015, β coefficient values of 0.7,0.79 and was trained on 400 epochs. Unsurprisingly, most hyperparameters that created the best performing models were similar, which provides confidence in their reliability especially the learning rate and hidden layer size of 15 appearing across all the models. Furthermore, the variance in accuracy across all models were low with the top and the worst performing models having an accuracy difference of 0.01.

SVM on the other hand, saw greater variance in model performance when utilising different parameters, with the worst performing model having an accuracy score of 0.537531 which is close to the probability from random chance for a binary problem.

4.2 Algorithm Comparison

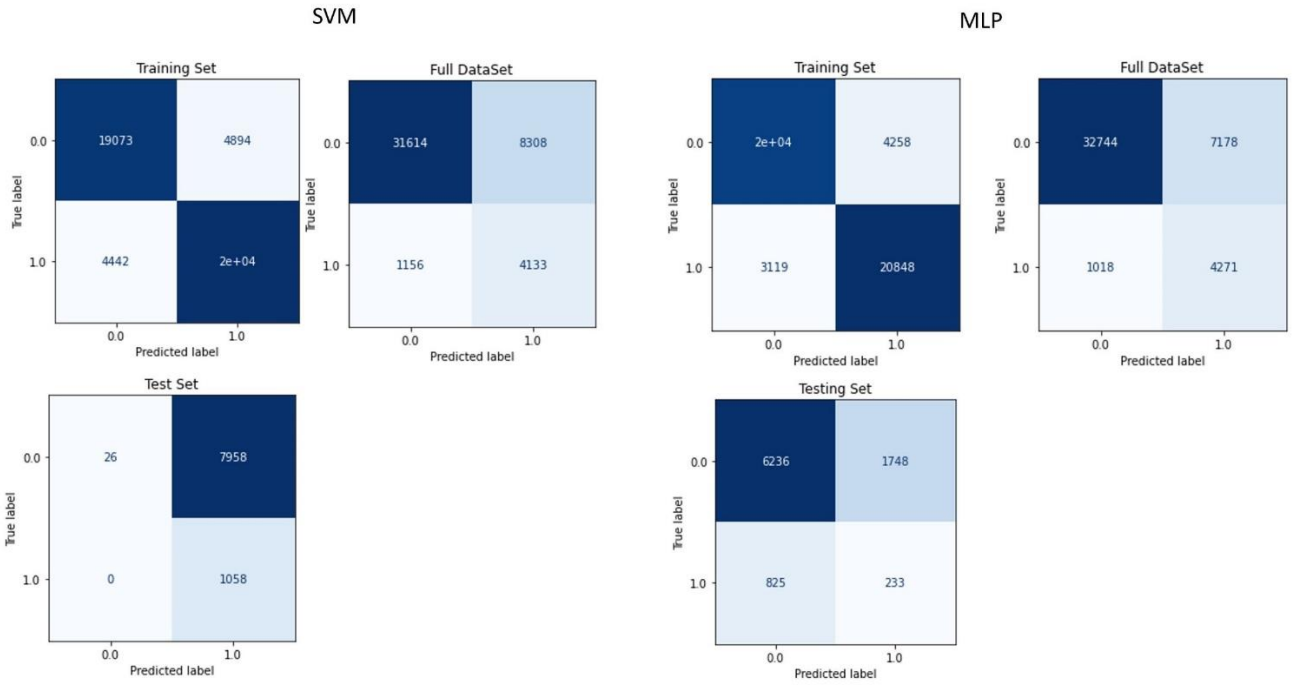

Figure 1. Confusion Matrix of model results.

Dataset	SVM	MLP
Training	0.8052	0.8461
Test	0.5016	0.5066
Full Dataset	0.7867	0.8139

Table 3. Accuracy results of our models on different datasets.

With reference to **Figure 1,** we see that both models had high levels of false predictions, with SVM seeing a misclassification rate of 99% whereas MLP only saw a misclassification rate of 32% on the test set, with the training set being 19% and 15% respectively. Looking at the confusion matrices, false positives were the most common misclassifications across both models, accounting for 98% of the misclassification errors seen on SVM and 20% for MLP.

To avoid the possibility of class imbalances affecting the scores each model would receive, the balanced accuracy score were used, however both models performed far worst on the testing set both seeing an accuracy close to 50% which is only slightly better than that of random chance. This could indicate that both models may have either overfitted to the training data or the class imbalances still have a significant impact, regardless of the different methods used to prevent these possibilities in both models.

Performance of both our models on the full dataset were higher than that of the testing set but this could be attributed to having seen over 60% of the dataset from the training set therefore the results are negligible. Overall MLP performed marginally better than SVM(as seen on **Table 3**) across all the metrics used which is in accordance with literature employing this dataset [5].

5. Conclusion

This paper compares the performance of the two algorithms MLP and SVM and their perform in predicting the success rate of marketing selling campaigns on individuals with varying backgrounds.

Performances between MLP and SVM are fairly similar across all tested metrics. It should be noted however that the performance of MLP is largely dependent on initial weight coefficients generated by the model which may result in high variance of model performance. Both models performed well on both the training and full datasets but significantly underperformed on the testing set. Overall MLP was identified as the better model for this problem as it managed to predict a large portion of the true positives while minimising the amount of false positives, which is important in this context as it would reduce the overall marketing cost incurred by providing a smaller more accurate pool of customers that would have a higher subscription rate

To provide a better comparison of both MLP and SVM, future work includes utilising a dataset with lesser degree of class imbalances with the utilisation of other metrics such as Precision, Recall and F1 Score which can give more insights into the accuracy of the model relative to the traditional classification accuracy employed in this paper. For MLP, a different early stopping criteria could be used e.g. instead of using validation score as the criteria, the increase of generalization error on the validation set could be used instead. In terms of SVM, performing parameter search of box constants with higher values should be considered which would provide stricter separations and thus reduce the possibilities of overfitting. Different training techniques could also be used such as bagging to decrease variance of our results and helps resolve over-fitting issues as well [6].

References

[1] S.S. Haykin.Neural networks and learning machines. Prentice Hall, 2009.

[2] C. Cortes and V. Vapnik. Support Vector Networks.Mach. Learning., 20(3):273–297, 1995

[3] Mojarad, S.A., Dlay, S.S., Woo, W.L. and Sherbet, G.V. Cross validation evaluation for breast cancer prediction using multilayer perceptron neuralnetworks. American Journal of Engineering and Applied Science., 2011.

[4] Andrew, A.M., 2000. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods by Nello Christianini and John Shawe-Taylor, Cambridge University Press, Cambridge, 2000, xiii+ 189 pp., ISBN 0-521-78019-5

[5] Moro, S., Cortez, P. and Rita, P., *A Data-Driven Approach to Predict theSuccess of Bank Telemarketing*., 2021

[6] Oza, N.C. and Russell, S., 2001, August. Experimental comparisons of online and batch versions of bagging and boosting. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 359-364).ACM.

Appendix 1 – Implementation details

- Models should be trained on the “train_balanced.csv” dataset
- Testing of the model would be done on the “test_set.csv” dataset
- Full Data model run should be conducted on the “bank_clean.csv” dataset

Initial construction of the model using “pytorch” was successful being able to train the dataset using the “dataloader” function. However, the integration with python’s “skorch” package proved to be a difficulty as “sklearn’s” scoring metrics did not work well with the tensor flow datatype. Attempts to use the “BCELoss” function as a metric whilst performing grid search for the MLP model as an alternative also proved to not be fruitful. Therefore, models were switched to being constructed using “sklearn” in-built functions.

Initially, the model was trained on the untransformed dataset which resulted in extremely poor performances across both models. However once the dataset was scaled, the results saw a significant improvement.