A Comparison of Multilayer Perceptron and Support Vector Machines on a Marketing dataset

Edward St John

edward.st-john@city.ac.uk

Abstract

This paper aims to provide a critical analysis and evaluation of two neural networks based on a bank marketing dataset to predict if customers will sign up to a deposit with the bank. Our two algorithms are Multilayer Perceptron (MLP) and Support Vector Machines (SVM). A number of models and hyperparameter combinations were analysed before the best models were evaluated on a test set of data using Recall and Receiver Operation Curves (ROC) along with the ROC area values (AUC). For our classification problem, SVM proved to be the most effective model.

1. Brief description and motivation of the problem

Mass advertising campaigns often have a very low success rate [1], whereas targeted campaigns often have a much higher yield. This is due to the ability to specify certain parameters when it comes to the population advertisers target, meaning they have a higher chance of converting those campaign attempts into sales (or in this case bank deposits) given that they know this particular set of attributes in their target population have a higher chance of depositing compared to the general population.

The aim of this paper is to compare and evaluate two different Neural Networks on the UCI Bank Marketing dataset [2],[3] where we try and predict if a customer will subscribe to a deposit or not. The Neural Networks in question are the Multilayer Perceptron and Support Vector Machines.

2. Description of the dataset including data types (discrete, continuous, etc.)

This Bank Marketing dataset from UCI contains a total of 4,521 samples with 16 different variables including 7 continuous (Table 1 below) and 9 categorical variables (Table 2).

Table 1 – Numerical variables overview

	count	***mean***	***std***	***min***	***25%***	***50%***	***75%***	***max***
age	4521	41.17	10.58	19	33	39	49	87
balance	4521	1422.66	3009.64	-3313	69	444	1480	71188
day	4521	15.92	8.25	1	9	16	21	31
duration	4521	263.96	259.86	4	104	185	329	3025
campaign	4521	2.79	3.11	1	1	2	3	50
pdays	4521	39.77	100.12	-1	-1	-1	-1	871
previous	4521	0.54	1.69	0	0	0	0	25

Table 2 – Categorical variables overview

	job	**marital**	**education**	**default**	**housing**	**loan**	**contact**	**month**	**poutcome**
count	4521	4521	4521	4521	4521	4521	4521	4521	4521
unique	12	3	4	2	2	2	3	12	4
top value	management	married	secondary	no	yes	no	cellular	may	unknown
frequency	969	2797	2306	4445	2559	3830	2896	1398	3705

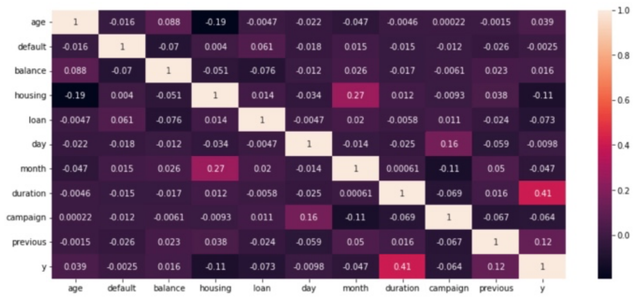
As you can see from the Table 1 above, there is a wide range of values across all the different numerical variables. If left as is, any model being trained on this data may attach more significance to a certain feature if it has high variance, giving it more weight in its final decision. This can lead to poor model performance due to that feature dominating the model decision more than it should. Scaling the data can help to overcome this issue, as well as decreasing training time by making the data much easier for the model to understand and process since it brings all the data points closer together.

In terms of categorical variables from Table 2, they all needed to be encoded in order for the models to be able to process them. One-Hot Encoding was used for all of them (splitting each variable value up into its own contained variable, then placing either a 1 or a 0 to indicate the feature being present or not) apart from the month variable where there may be significance with the time of year (e.g. later in the year being better for targeting customers) so it was handled with Label Encoding (just turning the months into numbers from 1 to 12).

There were other various data preparation tasks needed on this dataset such as dropping unknown entries so that it didn't affect the models or dropping features entirely due to too many unknown values leading them to be effectively dummy variables ('pdays' with 75% of values being -1 indicating no data collected and 'poutcome' having 75% of values unknown). Also, our data was fairly unbalanced with a large difference in samples for each class, so the SMOTE technique [4] was used to balance the data (just for the training set, the test set remained unseen as it was to prevent any bias).

Table 3 adjacent shows the correlation matrix of all the variables. As you can see, the 'duration' variable (the duration of each marketing call) seemed to correlate heavily with the target outcome. I concluded this variable should be dropped as for real world use of these models the duration of the call isn't something that is known before the marketing campaign commences, and so isn't considered appropriate to be included in our model training.

Table 3 – Correlation Matrix



	age	default	balance	housing	loan	day	month	duration	campaign	previous	y
age	1	-0.016	0.088	-0.19	-0.0047	-0.022	-0.047	-0.0046	0.00022	-0.0015	0.039
default	-0.016	1	-0.07	0.004	0.061	-0.018	0.015	-0.015	-0.012	-0.026	-0.0025
balance	0.088	-0.07	1	-0.051	-0.076	-0.012	0.026	-0.017	-0.0061	0.023	0.016
housing	-0.19	0.004	-0.051	1	0.014	-0.034	0.27	0.012	-0.0093	0.038	-0.11
loan	-0.0047	0.061	-0.076	0.014	1	-0.0047	0.02	-0.0058	0.011	-0.024	-0.073
day	-0.022	-0.018	-0.012	-0.034	-0.0047	1	-0.014	-0.025	0.16	-0.059	-0.0098
month	-0.047	0.015	0.026	0.27	0.02	-0.014	1	0.00061	-0.11	0.05	-0.047
duration	-0.0046	-0.015	-0.017	0.012	-0.0058	-0.025	0.00061	1	-0.069	0.016	0.41
campaign	0.00022	-0.012	-0.0061	-0.0093	0.011	0.16	-0.11	-0.069	1	-0.067	-0.064
previous	-0.0015	-0.026	0.023	0.038	-0.024	-0.059	0.05	0.016	-0.067	1	0.12
y	0.039	-0.0025	0.016	-0.11	-0.073	-0.0098	-0.047	0.41	-0.064	0.12	1

3. Neural network choice summaries

3.1 Multiplayer Perceptron (MLP)

MLP is a supervised learning classifier composed of an input layer, any number of hidden layers and then an output layer. The hidden layers are the part of the model where the main computations are done assigning different weight values to observations which are then passed through an activation function to obtain an output vector. Backpropagation [5] is used when the output and real target are compared creating an error, which is then passed back through the model to update the weights. MLP has very high predictive power and is particularly good at pattern recognition, however there are often issues arising from the large number of parameters needed and the difficulty in explaining the output given its black box characteristic.

3.2 Support Vector Machine (SVM)

SVM is also a supervised learning model praised for being one of the best ‘off-the-shelf’ algorithms. The concept is that it maps the input data assigning it to the different classes (0 and 1 in our case) on parallel hyperplanes trying to maximise the distance between the hyperplanes of both classes [6]. This leads to SVM being very effective in high dimensional spaces, however with large datasets proves to be a lot of work computationally compounded with its complex kernel.

4. Hypothesis Statement

Even though this is a challenging dataset with many different variables, both MLP and SVMs are good choices for classification tasks and so should perform fairly well. Evidence seems to suggest that MLP will perform slightly better when it comes to the prediction results compared to SVM [7], however we do expect an SVM to be able to train much faster than a MLP given it’s based on a local approximation strategy which is much faster than the global method used by MLP [8]. This would be significant for larger scale uses of these models and should be factored into a decision over which model is best suited for the task.

5. Description of choice of training and evaluation methodology

Our experiment methodology consists of holding 25% of the dataset as testing data. The other 75% is then split into training and validation data, with 25% of that portion being held for our validation data for our model selection.

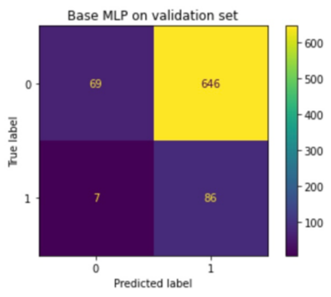
A pipeline was used in training of both the models when using our gridsearch technique to encapsulate both SMOTE and scaling in order to prevent any data leakage during the cross validation into the validation sets (which would have been the case if we scaled and applied SMOTE to the data beforehand). A test was performed with the baseline SVM model on our original data, but it performed extremely poorly with nearly every prediction being the negative class from not generalising well, so therefore SMOTE was used throughout.

Confusion matrices and classification reports (showing accuracy, precision, recall and f1) will be generated for each model, and then the best model will then be selected based on those results to be brought forward and used for testing.

6. Choice of parameters and experimental results

A grid search was used for both the SVM and MLP models to find their optimal hyperparameters based on the training data. For SVM, the hyperparameters searched were the Kernel function (Linear, Polynomial and Radial Basis), the regularisation parameter C (with different values tested), and the degree of the function (again with different values). For MLP, there hyperparameters searched were the number of neurons in the hidden layer, the activation function for the hidden layer (logistic, tanh and relu) and the learning rate.

Basic models were created without any hyperparameter tuning to establish a baseline metric to compare our optimised models to, using the same parameters as previous studies on this topic have used before [9]. The base MLP model seemed to have an excellent recall score, but on closer inspection with a confusion matrix (see picture adjacent) we can see it only achieved this high score as a result of most predictions being the same class which isn't particularly useful for us.



Accuracy was originally used as the scoring method for the grid searches, but the models kept defaulting to predicting everything as a single class in order to obtain a high accuracy (clearly not proving a useful model). Switching the scoring method to recall (a much better metric for us in this situation as we aim to minimise the number of successful deposits we miss) yielded much better results. As you can see in the table above however, there were some cases in the SVM search of terrible recall results where the model would have issues, with the linear kernel not being able to handle the data at all in some cases.

7. Analysis and critical evaluation of results

7.1 Model selection

Table 4 below shows the results of both grid searches for the models

Table 4 – Hyperparameter results

SVM			
Kernel	C	Degree	Score
linear	0.1	2	0.00714
poly	0.1	2	0.02321
rbf	0.1	2	0.00000
linear	0.1	3	0.00714
poly	0.1	3	0.04464
rbf	0.1	3	0.00000
linear	0.1	4	0.00714
poly	0.1	4	0.10089
rbf	0.1	4	0.00000
linear	1	2	0.00000
poly	1	2	0.01607
rbf	1	2	0.02500
linear	1	3	0.00000
poly	1	3	0.12679
rbf	1	3	0.02500
linear	1	4	0.00000
poly	1	4	0.23661
rbf	1	4	0.02500
linear	5	2	0.00000
poly	5	2	0.05982
rbf	5	2	0.17500
linear	5	3	0.00000
poly	5	3	0.27411
rbf	5	3	0.17500
linear	5	4	0.00000
poly	5	4	0.39286
rbf	5	4	0.17500

MLP			
Hidden Layers	Learning Rate	Activation	Score
50	0.05	logistic	0.15714
50	0.1	logistic	0.18571
50	0.3	logistic	0.12143
100	0.05	logistic	0.19286
100	0.1	logistic	0.22857
100	0.3	logistic	0.18929
200	0.05	logistic	0.21071
200	0.1	logistic	0.17857
200	0.3	logistic	0.11071
50	0.05	tanh	0.19286
50	0.1	tanh	0.15714
50	0.3	tanh	0.16071
100	0.05	tanh	0.20357
100	0.1	tanh	0.13571
100	0.3	tanh	0.20714
200	0.05	tanh	0.19643
200	0.1	tanh	0.29286
200	0.3	tanh	0.27500
50	0.05	relu	0.16786
50	0.1	relu	0.11786
50	0.3	relu	0.07857
100	0.05	relu	0.18214
100	0.1	relu	0.12500
100	0.3	relu	0.18571
200	0.05	relu	0.17857
200	0.1	relu	0.15714
200	0.3	relu	0.01429

The best performing model from the gridsearch SVM was a 4th degree polynomial with a C value of 5 (seen in Table 4 above), achieving a recall score of 0.31 on the validation set (with a training score of 0.39 seen above). However, the base SVM model we created actually performed better with a recall score of 0.53 surprisingly, so we ended up choosing the base model as the SVM model being brought forward for testing.

The best performing model for MLP had 200 hidden layers, a learning rate of 0.1 and a tanh activation function. It drastically outperformed the baseline MLP model used a as a comparison (seen in the picture above in Section 6) which implies that there was no way near enough hidden layers in order to properly understand the data and get a reasonable output, so the gridsearch focused on higher values. Initially the model didn't have enough iterations to be able to converge at all so the maximum number of iterations was increased to combat this (not by much though in order to avoid overfitting).

7.2 Analysis of results

Figure 1 – Test set confusion matrices

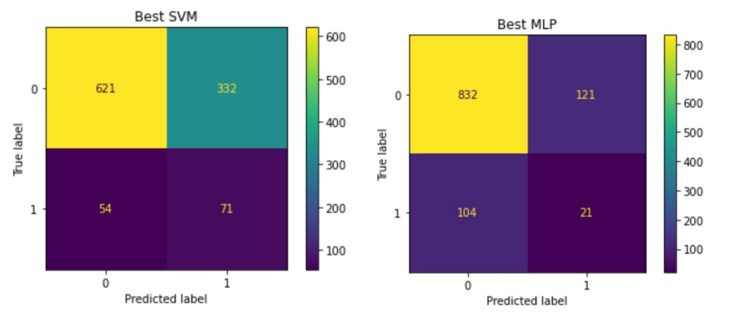


Table 5 – Best model test scores

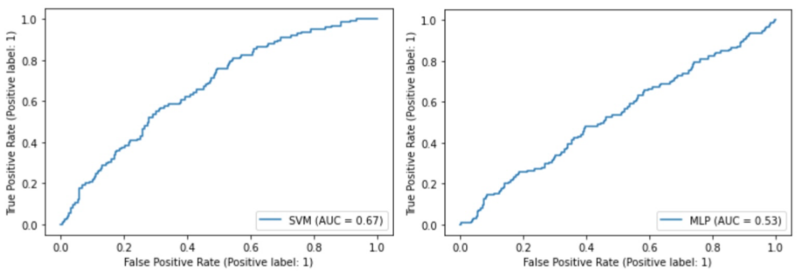
MLP			
	Precision	Recall	F1
0	0.89	0.87	0.88
1	0.15	0.17	0.16

SVM			
	Precision	Recall	F1
0	0.92	0.65	0.76
1	0.18	0.57	0.27

The confusion matrices for both models on the test data (which was previously unseen by both) can be seen above in Figure 1 as well as the scores adjacent in Table 5 (with class 0 being an unsuccessful call and 1 being a successful one). In our case of trying to predict conversions from a targeting marking strategy into deposits, recall is the most important metric for us. This is because we want to capture as many potentially successful calls as possible, and don't mind if there are some calls which don't end up being successful. Obviously, the bank could have a different strategy in mind and wanted to minimise the number of unsuccessful calls (in which case precision would be a more suitable measure of performance) but for the purposes of this paper we have assumed the above.

Overall, our optimal SVM model managed to achieve a much higher recall score than the optimised MLP model with a score of 0.57 to the MLP's 0.17. This shows the SVM doing a much better job at capturing more successful conversions with MLP missing 104 cases, a relatively large number considering the dataset properties. This was surprising given our hypothesis from Section 4 expecting MLP to outperform SVM for our testing but nevertheless it seems our baseline SVM model managed to generalise to the data far better than both the optimised SVM model and MLP and is yielding much better results.

Figure 2 – ROC curve and AUC scores



The ROC curve and AUC values are also useful metrics for evaluating models. The ROC curve displays the performance of the model at different classification thresholds, whereas the AUC score is simply the area below the curve providing an aggregate performance. Essentially, the higher the AUC score the better the model. While both models had fairly poor ACU scores as you can see in Figure 2 above, SVM did manage to outperform MLP by a decent margin (0.67 to 0.53). This again helps disprove our hypothesis from above and shows SVM to be the better performing model.

In terms of training time, our optimal SVM model (which was the baseline version rather than the model utilising grid search) took just a few seconds to train on the data given its simplicity (the grid search model took about a minute). The optimised MLP model however took nearly seven minutes to train, which will be due to the extra complexity of MLP compared to an SVM especially since it had so many neurons in the hidden layers to test the data through during the grid search process. This supports our hypothesis about the lower training time for the SVM model.

8. Conclusions, lessons learned and future work

This paper shows how well two trained neural networks of a Multilayer Perceptron (MLP) and a Support Vector Machine (SVM) were able to successfully predict a conversion of a customer into making a deposit with a bank through marketing calls, as well as critically analysing and comparing the two networks with each other.

We learnt that the tanh activation function was by far the most effective in the MLP model and had the most effect on the training scores, however it still wasn't good enough to beat the SVM when it came to testing them against each other. The SVM also managed to train extremely fast making it ideal for larger environments where there could be much more marketing data for the model to process.

For future work and if I had more time for this study, the exploration of adding a penalty to loss function in the training of the models would be interesting to study in terms of obtaining better performance through different penalties during training, assigning more weight to correctly predicted positive classes. Furthermore, adding some noise to the data before training the models could provide better generalisation which could again lead to better test performance on unseen data.

9. References

[1] Owen, L., 2000. Impact of a telephone helpline for smokers who called during a mass media campaign. Tobacco Control, 9(2), pp.148-154.

[2] S. Moro, P. Cortez and P. Rita, 2014. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, pp.22-31.

[3] UCI Machine Learning Repository: Bank Marketing Data Set. Available at: https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing (Accessed: 8th May 2022).

[4] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, 2002. SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, pp.321-357.

[5] Hecht-Nielsen, R., 1992. Theory of the backpropagation neural network. In Neural networks for perception, pp. 65-93.

[6] Suthaharan, S., 2016. Support vector machine. In Machine learning models and algorithms for big data classification, pp. 207-235.

[7] Osowski, S., Siwek, K. and Markiewicz, T., 2004. MLP and SVM networks-a comparative study. In Proceedings of the 6th Nordic Signal Processing Symposium, pp. 37-40.0

[8] Trenn, S., 2008. Multilayer perceptrons: Approximation order and necessary number of hidden units. IEEE transactions on neural networks, 19(5), pp.836-844.

[9] Yu, JM., Cho, SB., 2016. Prediction of Bank Telemarketing with Co-training of Mixture-of-Experts and MLP.

Appendix

Glossary

MLP: A Multilayer Perceptron is a type of feedforward neural network utilising hidden layers so solve pattern recognition problems

SVM: A Support Vector Machine is a supervised machine learning algorithm used in pattern recognition

Training set: The set of observations used to train and generate models

Test set: The set of observations used to evaluate a model's performance after model training has taken place

SMOTE: Synthetic Minority Oversampling Technique – a technique used to oversample the observations in the minority class to create a more balanced dataset

Generalisability: How well a model is able to adapt to new and unseen datasets

C: The parameter controlling the strength of regularisation in a SVM model

Learning rate: The rate at which the weights of a MLP model update

Confusion matrix: A table displaying the performance of models split up by true and false positives and negatives

Accuracy: The percentage of predictions that a model classified correctly

Recall: A measure showing the proportion of actual positive cases correctly identified by a model

Precision: A measure showing the proportion of correct positive predictions from a model compared to the total amount of positive predictions

Implementation details

The functions used for training the models were as follows:
sklearn.svm.svc

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

sklearn.neural_network.MLPClassifier

- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

sklearn.grid_search.GridSearchCV

- https://scikit-learn.org/0.15/modules/generated/sklearn.grid_search.GridSearchCV.html#sklearn.grid_search.GridSearchCV

The SMOTE technique [4] was applied to the data (either beforehand for baseline models or in a pipeline during the grid search process for the optimised models) using the imblearn sampling library in Python

SVM_Best.joblib and MLP_Best.joblib are our optimal SVM and MLP models used for the testing stage (which are automatically created if the full notebook is run from the start rather than just the testing stage).

X_test.csv and y_test.csv are the test sets to be used when evaluating the models.

The code has been tested in an environment running Python 3 (3.7.13) with the following packages installed: *pandas, numpy, matplotlib, imblearn, sklearn, joblib.*