INM427 Neural Computing Individual Project
Yumi Heo (Msc Data Science / 230003122 / yumi.heo@city.ac.uk)

A Comparison of Multilayer Perceptrons and Support Vector Machines for Bank Churn Prediction

1. Description and Motivation

Analysing customer churn in the bank is important for maintaining profitability, customer satisfaction, and competitiveness against other banks. Therefore, banks must enhance their ability to identify potential customer churn. This can be achieved using a supervised classification model based on neural networks or support vector machines (SVM), using customer and churn data. Multilayer perceptrons (MLP) and SVM have been utilised to conduct this experiment to find which of the two models better performs in classifying customer churn. By comparing and evaluating the performance of the two models, we will learn about the characteristics of the two algorithms and determine which model has the higher accuracy for this task and which model is appropriate for this case. Eventually, the best model will enable banks to predict future churn from customer data and proactively improve services to retain their customers.

2. Initial Analysis of Dataset

This bank churn dataset was obtained from Kaggle [1]. It is a tabular dataset which has 165,034 observations with 14 variables, which are 13 features and 1 target. The target is a column named ‘Exited’, which is binary. In the target, '1' means that a customer closed their account, and '0' means that a customer kept their account. The target is imbalanced since customer churn is 1/4 the rate of customer retention. Therefore, to resolve target imbalance, oversampling, undersampling, or stratified k-fold cross-validation can be considered before model training.

There is a large proportion of '0' in the feature variable 'Balance'. It should be examined whether to remove the data with '0' in 'Balance' or not, as this data might affect the sensitivity of the class distribution. After checking the histogram of credit score and the target variable, a similar distribution has been found in a dataset containing only '0' in ‘Balance’ and a dataset without '0' in ‘Balance’ (Figure 1). This implies that the dataset with '0' in 'Balance' is not significantly different from the dataset without '0' in 'Balance' in terms of its impact on the target.

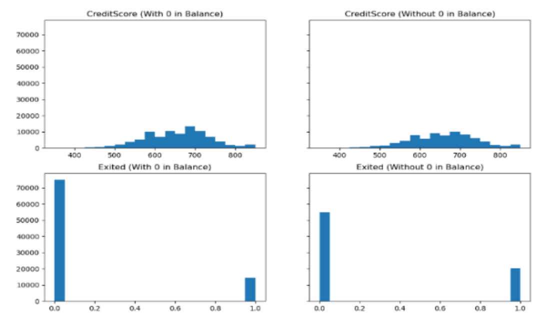

Figure 1. Histograms of credit score and the target

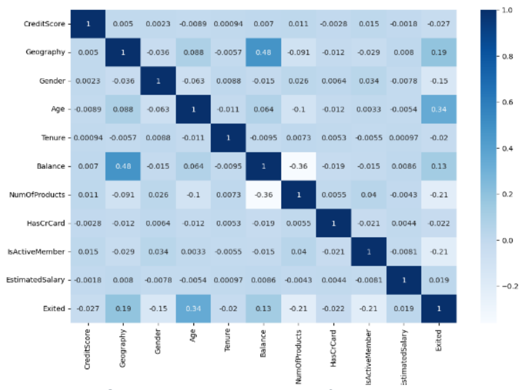

Figure 2. Correlation between all features and the target

Therefore, the data with the value of 0 in 'Balance' has been retained. The features 'id', 'CustomerId', and 'Surname' have been removed as they are irrelevant to predicting whether a customer will close their account. To check the correlation coefficients between variables, Pearson correlation was used (Figure 2). The highest correlation coefficient is 0.48, indicating that all features have low correlations with each other. Hence, 10 features have been used for modeling without any additional feature deletions.

Finally, considering the differing scale range of each feature, the data has been scaled between 0 and 1 after splitting it into train, validation, and test sets. This ensures stable results during the training of the MLP [2] and SVM models [3].

3. Hypothesis Statement

Before comparing the two different models, the following three hypotheses were developed:

1) SVM takes longer to train than MLP because it is unsuitable for running multiple samples [4].
2) The accuracy of the SVM final model will be higher than that of MLP because SVM shows higher performance in classification tasks than MLP [5].
3) The SVM model will provide a more interpretable decision boundary using support vectors.

These three hypotheses will be confirmed through the process and results of the experiment.

4. Summary of the MLP and SVM with their pros and cons

4.1. MLP

MLP is a supervised learning method and one of the neural networks used to address and solve non-linear problems [6]. This type of neural network is feedforward, consisting of fully connected layers with more than one hidden layer and units, the number of which can be controlled.

4.1.1. Pros

- MLP learns and provides the result from various types of data such as tabular or time-series data, images or text [7].
- MLP is used to solve classification or regression tasks [8].

4.1.2. Cons

- There's a risk of overfitting when the model contains a large number of weights, which can lead to it fitting the training data too closely [8].
- Determining and monitoring the values of weights is challenging [8]. If they are close to 0, the model may not learn effectively from the data, while excessively large weights can lead to poorer training performance.
- The interpretability of this model's reasoning is challenging due to its nonlinear activation functions and the opacity of its weights.

4.2. SVM

SVM is a supervised learning method, based on statistical learning principles [9]. This method effectively addresses both linear and non-linear problems through the use of kernel tricks, which transform input data into higher-dimensional space to tackle the task.

4.2.1. Pros

- SVM effectively captures linear or non-linear relationships in data by mapping it to higher dimensions, making it suitable for tasks with texts or images.
- It can be used for both classification [10] and regression [11] tasks.

4.2.2. Cons

- SVM is computationally expensive and impractical when dealing with large datasets because it calculates the kernel as a matrix, requiring significant memory storage [12].
- This method is sensitive to variations in feature scales, which can negatively affect its performance [3].

5. Methodology

All data used for developing the MLP model was converted into PyTorch tensors, while for the SVM model, the training and test sets in tensor state were converted to arrays using new variables. The original dataset was divided into 80% for training and 20% for testing. To handle the class imbalance issue, repeated stratified K-fold cross-validation was employed for model training and validation. This process repeats the cross-validation with different randomisations multiple times, ensuring each class's ratio is maintained. The training set was divided into three sets, creating a validation set using 20% of the training set, and the process was repeated twice, resulting in a total of 6 different models fitted and evaluated.

Subsequently, baseline models were created for each method. During training and evaluation, both models used normalised data, as both algorithms require data scaling. Normalisation was applied to the training set samples, while validation and test samples were transformed accordingly.

Following normalisation, hyperparameter tuning was performed through grid search or manual search. The best model was selected based on test accuracy and other evaluation metrics such as precision, recall, and F1 score during hyperparameter tuning.

Finally, the comparison of the best models from both algorithms was examined through evaluation metrics such as test accuracy, ROC curve and Precision-Recall curve.

6. Choice of Parameters and Experimental Results

As the target classes are imbalanced, repeated stratified K-fold cross-validation was applied during training on both MLP and SVM models. The losses of MLP models created through this process were combined and averaged, as were the accuracies of SVM models. After this training and validation process with hyperparameter tuning, trained models were compared with the test accuracy using the holdout test set.

6.1. The structure and parameters of the MLP model

For the activation function, ReLU(Rectified linear unit) was used in the hidden layer, and the sigmoid function was used in the output layer because the two classes should be classified. The optimizer was Adam, and the activation function and optimizer were fixed in this study. The loss function was set as BCELoss(Binary Cross Entropy) provided by PyTorch. Batch size can be used as a parameter but it was fixed as 64 for consistent experiment. The MLP model started with hyperparameter tuning in one hidden layer and progressed to two hidden layers. The case of three hidden layers was not tested in this study because it is used when dealing with complex or high-dimensional data [13]. However, the number of hidden units, learning rate, weight decay, and epochs were changed and tested. The best model was selected by comparing the test accuracy.

Multilayer Perceptrons							
No.	**Hidden layers**	**Hidden units**	**Epochs**	**Learning rate**	**Weight decay**	**Test Accuracy (%)**	**Training & Validation Time (sec)**
Baseline (one single hidden layer)							
1	1	6	50	0.001	0	86.28	441.65
Grid search regarding hidden units & learning rate							
2	1	6	10	0.001	0	85.86	89.02
3	1	4	10	0.0001	0	83.55	86.78
Grid search regarding weight decay							
4	1	6	10	0.001	0	85.87	88.86
5	1	6	10	0.001	0.0001	85.88	89.38
Application of linear learning rate scheduler							
6	1	6	10	0.0001→ 0.001	0.0001	86.20	201.99
Baseline (two hidden layers)							
7	2	(4, 4)	50	0.001	0	86.47	499.48
Grid search regarding hidden units & learning rate							
8	2	(6,4)	12	0.01	0	86.31	119.94
Increase the epochs							
9	2	(6,4)	20	0.01	0	85.89	199.45
Grid search regarding hidden units & learning rate							
10	2	(4, 4)	12	0.001	0	86.28	283.89
11	2	(4, 4)	12	0.01	0	85.66	271.33
Grid search regarding weight decay							
12	2	(6,4)	12	0.01	0.1	78.93	283.90
13	2	(6,4)	12	0.01	0.001	86.28	293.12

Application of linear learning rate scheduler							
14	2	(6,4)	12	0.001→ 0.01	0	86.32	263.23

Table 1. Hyperparameter tuning in MLP models

First, the experiment was initiated by randomly setting the number of hidden units and epochs and applying a learning rate of 0.001. Using the grid search provided in Scikit-learn and the grid search implemented using nested for loops, the test accuracy for each resulting value was calculated. Weight decay was obtained and tested through grid search, but it was revealed that the final model achieved the highest accuracy without weight decay. The final model, which emerged after applying a learning rate scheduler, consisted of two hidden layers with 6 and 4 hidden units, respectively. To prevent overfitting, 12 epochs were applied, and a learning rate that increased from 0.001 to 0.01 was used (Table 1).

6.2. The structure and parameters of the SVM model

The main kernel used in this experiment was the RBF (radial basis function) kernel. For the regularisation parameter, C was controlled and tested to maximise or minimise the margin of classification error. Gamma, which is related to the RBF kernel, was tested to observe its influence on the decision boundary. Additionally, the maximum number of iterations was adjusted and tested to ensure model convergence.

Support Vector Machines						
No.	**Kernel**	**C**	**Gamma**	**Iteration**	**Test Accuracy**	**Training & Validation Time (sec)**
Baseline						
1	rbf	1.0	scale	49821	85.71	1149.42
Manual search						
2	rbf	1.0	scale	10000	56.51	656.45
3	rbf	1.0	scale	5000	47.47	395.31
4	rbf	1.0	scale	1000	73.20	156.19
5	rbf	1.0	auto	25112	84.57	2155.29
Halving grid search						
6	rbf	100	0.1	460215	85.73	1352.12

Table 2. Hyperparameter tuning in SVM models

As training and grid search for SVM models required a large amount of time, manual search was conducted initially to assess each accuracy from the search. Firstly, training and validation times were calculated while adjusting the iteration, and it was found that even when the iteration was reduced from around 50,000 to 1,000, the level of accuracy was suitable for testing. Additionally, the accuracy was evaluated by changing the type of gamma without changing the iteration. A halving grid search provided by Scikit-learn was applied since grid search was computationally expensive. It selects the best parameter candidates with fewer resources. For reference, this estimator is still being tested according to the explanation from Scikit-learn. The best model was obtained by applying the parameters obtained through halving grid search, with a C value of 100 and a gamma value of 0.1 (Table 2).

7. Analysis and Critical Evaluation of Results

Training and grid search for the SVM model takes more time than the MLP model. When comparing the elapsed time on training of the best models, there is a difference of more than 5 times between SVM and MLP. It is not clear how much data is considered large for SVM, but a total of 132,027 training and validation data points were large in this case.

Although we assumed the accuracy of the SVM final model would be higher than that of MLP because SVM showed higher performance in classification tasks than MLP [5], the result was the opposite. The dataset may have been better suited to the MLP than SVM, and the complex non-linearity relationship inherent in the dataset may have made it difficult for SVM to classify compared to MLP. Even if repeated stratified K-fold cross-validation was applied, the performance of SVM may have been relatively lower because the dataset was not completely balanced.

Nevertheless, the SVM model would provide a more interpretable decision boundary using support vectors. Since we can find the number of support vectors, this model can relatively explain why such a boundary was set when compared to MLP. On the other hand, MLP is difficult to explain why such weight and bias calculations are made, how, and what made MLP decide to classify a data point to a certain class.

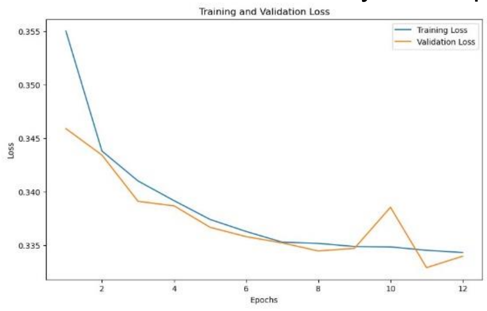

Figure 3. Loss plots of the final MLP model

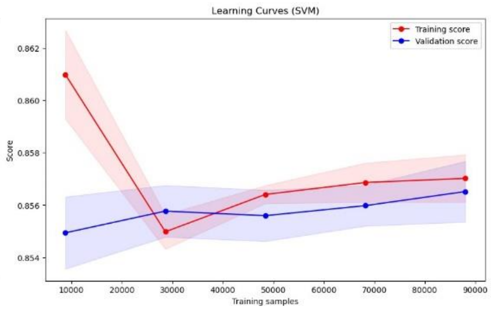

Figure 4. Learning curves of the final SVM model

Comparing the training process of the final MLP and SVM model, both plots (Figure 3 & 4) indicate those models were learning effectively as the score from the validation set was mostly lower than that of training. However, the small fluctuations in those two plots suggest some degree of overfitting. Lastly, the performance of the two models was compared through test accuracy, ROC curve, and Precision-Recall curve.

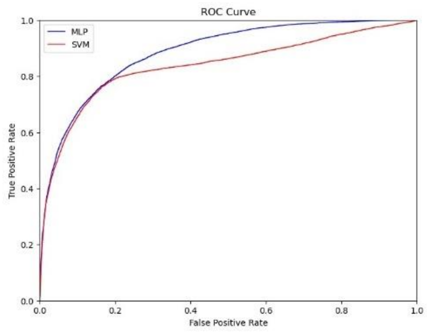

Figure 5. ROC Curve

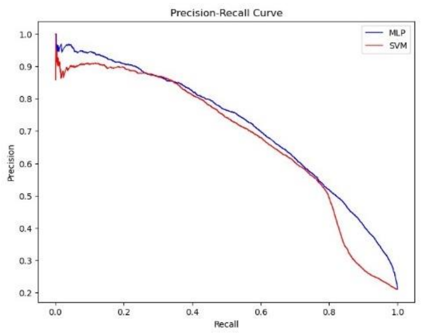

Figure 6. Precision-Recall Curve

In terms of test accuracy, the MLP model achieved 86.32% and 85.73% for the SVM model. The ROC curve and Precision-Recall curve of MLP also showed more stable performance, achieving 0.89 for AUC and 0.72 for Average Precision score. This indicates that the MLP model is a better method for solving the classification problem of the dataset used in this study.

8. Lessons Learned and Future Work

Depending on the dataset, the training speed and accuracy of MLP and SVM may differ, and it is difficult to say which one generally shows higher accuracy and faster speed. Here, MLP shows superior speed and accuracy for the relevant dataset. However, when used in business that requires an explanation of the model, MLP should be considered carefully for use as it is difficult to interpret its performance.

For future work, more detailed work would be done through EarlyStopping provided by PyTorch to prevent overfitting, rather than manually setting the epoch. it would be considered to further investigate the relationship between iteration and accuracy when training SVM. Also, applying additional class weights to SVM is one of the methods to solve the imbalance problem that was not solved by repeated stratified K-fold cross-validation. It is also a good idea to use an ensemble technique that combines the two models by extracting features using MLP and classifying data using SVM to solve other classification tasks [14].

Reference

[1] W. Reade and A. Chow, ‘Binary Classification with a Bank Churn Dataset’. Kaggle, Jan. 02, 2024. Accessed: Feb. 05, 2024. [Online]. Available: https://www.kaggle.com/competitions/playground-series-s4e1

[2] Scikit-learn: Machine Learning in Python. ‘Neural network models (supervised), Tips on Practical Use’. *scikit-learn*. Accessed: Apr. 16, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[3] Scikit-learn: Machine Learning in Python. ‘Support Vector Machines, Tips on Practical Use’. *scikit-learn*. Accessed: Apr. 16, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[4] S. Abe, Support vector machines for pattern classification. in Advances in pattern recognition. London: Springer, 2005.

[5] S. Osowski, K. Siwek, and T. Markiewicz, ‘MLP and SVM Networks – a Comparative Study’. Proceedings of the 6th Nordic Signal Processing Symposium., pp. 37-40, Jun. 2004.

[6] G. Cybenko, ‘Approximation by superpositions of a sigmoidal function’, *Math. Control Signal Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, doi: 10.1007/BF02551274.
[7] J. Brownlee, ‘When to Use MLP, CNN, and RNN Neural Networks’, Machine Learning Mastery. Accessed: Apr. 16, 2024. [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

[8] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Second edition, Corrected at 12th printing 2017. in Springer series in statistics. New York, NY: Springer, 2017. doi: 10.1007/b94608.

[9] V. N. Vapnik, *The nature of statistical learning theory*, 2nd ed. in Statistics for engineering and information science. New York: Springer, 2000.

[10] C. Cortes and V. Vapnik, ‘Support-vector networks’, *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[11] V. N. Vapnik, ‘The Support Vector method’, in *Artificial Neural Networks — ICANN’97*, vol. 1327, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds., in Lecture Notes in Computer Science, vol. 1327. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 261–271. doi: 10.1007/BFb0020166.

[12] J. Cervantes, X. Li, W. Yu, and K. Li, ‘Support vector machine classification for large data sets via minimum enclosing ball clustering’, *Neurocomputing*, vol. 71, no. 4–6, pp. 611–619, Jan. 2008, doi: 10.1016/j.neucom.2007.07.028.

[13] J. Heaton, ‘The Number of Hidden Layers’, Heaton Research. Accessed: Feb. 16, 2024. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html

[14] M. Yousefnezhad, J. Hamidzadeh, and M. Aliannejadi, ‘Ensemble classification for intrusion detection via feature extraction based on deep Learning’, *Soft Comput*, vol. 25, no. 20, pp. 12667–12683, Oct. 2021, doi: 10.1007/s00500-021-06067-8.

Appendix Ⅰ – Glossary

Term	Definition
MLP	Multilayer Perceptrons.
SVM	Support Vector Machines.
Observation	A single data point in a dataset.
Feature	An input variable to make a model.
Target	A variable that a model will predict.
Variable	Here, this is an independent variable that can be controlled.
Balance	A feature in the dataset. This is a continuous scaled value since it is the balance of a bank account.
id	A feature in the dataset. This is the index of the dataset.
CustomerId	A feature in the dataset. This is the ID given to each customer.
Surname	A feature in the dataset. This is the last name of each customer.
ReLU	Rectified linear unit.
Adam	An adaptive optimizer that combines the momentum and RMSprop.
RBF	Radial basis function.
ROC	Receiver operating characteristic.
AUC	Area under the ROC Curve.

Appendix Ⅱ – Implementation Details

In the case of MLP, many parameters could be adjusted so that more experiments were conducted compared to SVM. Among them, the one tested with manual search was the single hidden layer model, and the test accuracy was confirmed by randomly determining the number of hidden units. However, since only this parameter was adjusted, it was difficult to interpret the relationship between the trained model and test accuracy. As a result, the optimal learning rate and weight decay were discovered through grid search. Additionally, the learning rate scheduler was used to check test accuracy after the learning rate gradually decreased or increased. At first, it was assumed that test accuracy would increase when the learning rate was gradually reduced. Instead, however, when starting from a low learning rate and reaching a high learning rate, the model's test accuracy was slightly higher than that of other models. Furthermore, in the case of models 6 and 7 (Table 3), even though other parameters were the same, and the final learning rate was the same, the test accuracy varied greatly depending on the step reduction when different learning rate schedulers were applied.

Multilayer Perceptrons							
No.	**Hidden layers**	**Hidden units**	**Epochs**	**Learning rate**	**Weight decay**	**Test Accuracy**	**Training & Validation Time (sec)**
Manual Search							
1	1	4	10	0.001	0	86.19	87.15
2	1	5	10	0.001	0	85.74	87.31
3	1	2	10	0.0001	0	85.60	86.92
Application of step learning rate scheduler							
4	1	6	10	0.001→ 0.0001	0.0001	85.44	207.54
Application of exponential learning rate scheduler							
5	1	6	10	0.001→ 0.0005	0.0001	86.28	205.29
Application of step learning rate scheduler							
6	2	(6,4)	12	0.01→ 0.0048	0	78.93	273.42

Application of exponential learning rate scheduler							
7	2	(6,4)	12	0.01→ 0.0048	0	86.31	271.62

Table 3. Hyperparameter tuning in MLP models

The training speed of SVM was at least three times different from that of the MLP model, and it was difficult to derive results even when applying grid search only for kernel selection. As a result, testing for each kernel was done manually. The RBF kernel showed the highest accuracy, and the remaining kernels showed relatively low accuracy or, in the case of the sigmoid kernel, the lowest validation accuracy of 68% (Table 4). The training took quite a significant amount of time when using the SVC code from Scikit-learn. Hence, the LinearSVC code, also provided by Scikit-learn and suitable for large datasets, was used (Table 5). LinearSVC also has parameters that can be adjusted. Manual and grid searches were applied, and it was faster than when using SVC code for modeling.

Support Vector Machines (with the code ‘SVC’)					
No.	**Kernel**	**C**	**Gamma**	**Validation Accuracy**	**Training & Validation Time (sec)**
Manual search					
1	linear	1.0	scale	82.97	723.47
2	poly	1.0	scale	85.46	925.50
3	sigmoid	1.0	scale	68.10	1144.60

Table 4. Hyperparameter tuning in SVM models (with the code ‘SVC’ from Scikit-learn)

Linear Support Vector Classification (with the code ‘LinearSVC’)						
No.	**penalty**	**loss**	**C**	**max_iter**	**Validation Accuracy**	**Training & Validation Time (sec)**
Baseline						
1	l2	squared_hinge	1.0	1000	82.87	6.82
Manual search						
2	l1	squared_hinge	1.0	1000	82.87	4.12
3	l2	hinge	1.0	1000	83.06	1.41
Grid search & Manual search						
4	l2	squared_hinge	0.01	1000	82.80	1.32
5	l2	hinge	0.01	1000	79.41	1.32
6	l2	squared_hinge	0.001	1000	82.30	0.95

Table 5. Hyperparameter tuning in SVM models (with the code ‘LinearSVC from Scikit-learn)

In the case of LinearSVC code, the maximum iteration is already set to 1,000. Nevertheless, the accuracy was almost similar to that when using SVC's RBF kernel, which was over 80%, but showed a slightly lower level of accuracy. Likewise, the regularisation C can be adjusted, and the loss function can be selected between the hinge, commonly used in SVM models, or the squared hinge. However, although it was validated by applying the parameters obtained from the grid search and manual search during this experiment, it showed lower accuracy than the model established as a baseline. Overall, the models from this code were not considered significantly due to their lower accuracy compared to the RBF kernel of the SVC code. However, if fast speed for training and testing is more required by trading off accuracy, it may be considered to use LinearSVC.