A Comparison of Multilayer Perceptron and Support Vector Machine on Classifying Contraceptive Methods

Jessica Guan

Abstract - This paper explores the application of a multilayer perceptron and a support vector machine to solve a multiclass classification problem. The models will be used to classify women's contraceptive methods based on socioeconomic factors. Then, the models will be compared and analysed.

1. Introduction

Studies have linked birth control access to a higher quality of life. Research findings suggest that communities with higher rates of contraceptive usage experience lower crime rates. The authors further hypothesise that the correlation could be due to increased educational opportunities, financial opportunities, and a reduced number of unplanned pregnancies (N. Hill et al). However, communities in lower socioeconomic classes tend to have less access to contraceptive methods due to lack of healthcare and fewer contraceptive options. Exploring which socioeconomic factors most affect women's access to birth control and family planning is important in knowing what resources are needed in these communities. This research paper uses neural computing methods to classify women's contraceptive methods based on socioeconomic factors.

2. Exploratory Data Analysis

The dataset, consisting of 1473 rows and nine features, surveyed women in Indonesia about their contraceptive methods. The nine features are the wife's age, wife' education, husband's education, number of children, wife's religion, if the wife is working, husband's occupation, standard of living index, and media exposure. There are three contraceptive class options: 1 (no use), 2 (long-term use), and 3 (short-term use). As shown in Figure 1, the data was slightly imbalanced, so I tested the models with and without SMOTE. The results were similar, so the data without SMOTE was used. To increase accuracy in the neural network models, feature selection was performed. As shown in Table 1, the chi-square and p-value were calculated for each feature in relation to the target classes. These values were examined and used to determine if there was a significant correlation.

Figure 1: Class Counts before SMOTE

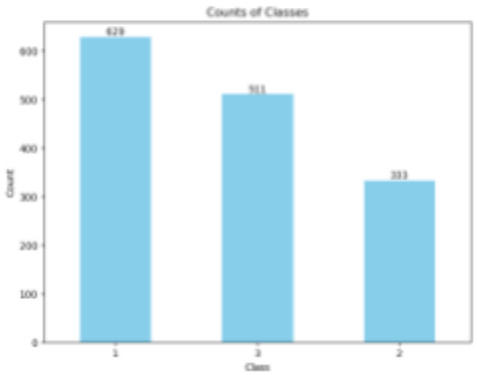


Table 1: Statistics of features

	Chi-square	p-value	Degrees of freedom
Wife age	160.86	6.88e-10	66
Wife education	140.56	8.02e-28	6
Husband education	73.95	6.0e-14	6
Number of children	219.09	1.56e-31	28
Wife religion	21.62	2.02e-5	2
Wife working	5.19	0.07	2
Husband occupation	65.4	3.57e-12	6
Standard of living index	62.2	1.61e-11	6
Media exposure	31.57	1.39e-7	2

3. Algorithms

3.1. Multilayer Perceptron (MLP)

A multilayer perceptron is a neural network that consists of an input layer, one or more hidden layers, and an output layer using feedforward. Each layer consists of a number of neurons. In the input layer, the number of neurons represents the number of features. The computation happens in the hidden layers with an activation function. The number of hidden neurons, type of activation function,

and number of hidden layers can be changed in hyperparameter tuning. Finally, the output of the computation is passed to the output layer, which gives the final result. The number of neurons in the output layer depends on the task. For example, the task in this study is multiclass classification with three classes, so the output layer has three hidden neurons (M.-C. Popescu et al).

3.1.1. MLP Pros

MLPs are widely used neural networks due to their versatility and scalability. MLPs can handle a variety of tasks including classification, regression, clustering, and dimensionality reduction. Additionally, because of the number of layers and neurons can be adjusted, MLPs are flexible models that can be tuned to the task at hand. It can also handle large datasets by increasing the layers and neurons. MLPs are also easily accessible as they are supported by many libraries such as PyTorch.

3.1.2. MLP Cons

MLPs can overfit when using high dimensional data, when they memorise the training data, or when dealing with small datasets. In overfitting, MLPs fail to capture the pattern in the data. They are also more accurate when dealing with a lot of data, so when the dataset is small MLP may not be the best choice. They can also be computationally intensive as the number of layers and neurons increase. Lastly, MLPs have several parameters that the user must evaluate.

3.2. Support Vector Machine (SVM)

A SVM is a machine learning algorithm that calculates the optimal hyperplane to divide and classify data by maximising the margin between the data. Although originally created for binary classification, SVMs can also be applied to regression and multiclass classification tasks using One-vs-One (OvO) or One-vs-Rest (OvR). In OvO, the SVM trains a binary classifier for each combination of two classes, while multiple classifiers are trained in OvR. SVMs have a number of hyperparameters including kernel type and regularisation. The kernel type transforms the input data into a higher dimensional space where the hyperplane can be more easily found. Regularisation determines the tradeoff between maximising the margin and minimising the error. For example, a smaller regularisation value allows for more errors, but a larger regularisation value gives a bigger penalty to errors.

3.2.1. SVM Pros

SVM is a robust machine learning algorithm. The regularisation hyperparameter protects against overfitting. It is also efficient in handling high-dimensional data due to the kernel trick. Unlike MLP, SVMs are robust in handling smaller datasets as well.

3.2.2. SVM Cons

SVMs are not as suitable as MLPs in handling large datasets. They also require more memory for storing support vectors, and SVMs can be computationally intensive.

4. Methods

4.1. Implementation

The SVM and MLP models were implemented in Python, using the libraries scikit-learn and PyTorch respectively. The data was split into 80% training data and 20% testing data. In each model, grid search was implemented to find the best hyperparameters, and k-fold cross-validation with five folds was used to assess the performance. K-fold cross-validation is a robust validation technique regardless of the dataset size. Because we are dealing with a relatively smaller dataset, k-fold was used. During k-fold validation, the training data was split into a training set and validation set. The training set was used for hyperparameter tuning and training, while the validation set was used to evaluate the hyperparameters and the model during training.

4.2. MLP Parameters

The parameters for the MLP model were chosen through grid search. Grid search iterated through the hidden neuron size, learning rate, and weight decay for L2 regularisation. I followed the guidelines provided by J. Heaton to determine the number of hidden neurons, stating that the number should be within the range of the size of the input layer and the output layer, two thirds of the size of input layer plus the size of the output layer, and less than twice the size of the input layer (J. Heaton, 2018). However, these numbers did not give high accuracy results. After further experimenting in grid search, the hidden neuron sizes 70 for the first hidden layer and 90 for the second hidden layer gave the best results. Using 5-fold cross-validation in grid search, the best parameters were found. However, the number of epochs, activation function, optimizer, and number of hidden layers were determined manually. 500 epochs showed the best accuracy, while any number higher or lower showed a slight decrease in accuracy. After testing Sigmoid, ReLu, ELU, and Tanh activation functions, the ReLu activation function yielded the best results. Both SGD and Adam optimizers were tested, and the Adam optimizer performed better, as it also integrates momentum.

Table 2: MLP parameters

Epochs	Hidden neuron size 1	Hidden neuron size 2	Hidden layers	Learning rate	Weight decay	Activation function	Optimizer
500	70	90	2	0.01	0.001	ReLu & Softmax (last layer)	Adam

4.3. SVM Parameters

The parameters for the SVM model were also chosen through grid search. Grid search iterated through different values for regularisation (C parameter), kernel type, the kernel coefficient (gamma), and class weights. Like for the MLP mode, 5-fold cross-validation and accuracy were used to find the best parameters.

Table 3: SVM parameters

C	Kernel	Gamma	Class weights
10	Radial basis function	1	Balanced

5. Results and Analysis

Table 4: Validation results

MLP		SVM
0.62	**Precision**	0.76
0.63	**Recall**	0.72
0.62	**F1**	0.73
0.64	**Accuracy**	0.73

Table 5: Test results

MLP		SVM
0.62	**Precision**	0.99
0.61	**Recall**	0.99
0.61	**F1**	0.99
0.63	**Accuracy**	0.99

SVM performed 14.06% better than MLP in accuracy on the validation data. As shown in Table 4, MLP had an accuracy rate of 0.64, while SVM had an accuracy rate of 0.73. In addition, SVM performed 57.14% better than MLP in accuracy on the test data. In Table 5, we can observe that SVM performed significantly better with an accuracy rate of 0.99, while MLP had an accuracy rate of 0.63 for the testing data. The precision, recall, and F1 results of SVM and MLP in both training and testing yielded similar metrics in which SVM noticeably showed better scores on the test data. The two models can be further compared using the visualisation of the confusion matrices for both the validation and test sets in Figure 2 and Figure 3. The SVM model showed a significant improvement on the test data than the validation data. Although discrepancy between validation and test data is to be expected, this is a large increase in performance results which requires careful investigation. I tested the results on consistency across several runs, and evaluated the training and testing procedures. Each run showed consistent results, and I could not find meaningful errors in the evaluation procedures. Another factor to consider is data leakage in which the test data could have been previously seen by the model. However, I ensured that the data was split into training and testing sets before hyperparameter tuning and training the model. Only the training data was used for hyperparameter tuning and training with k-fold validation. For training speeds, the SVM model had an average training speed of 0.9 seconds, and the MLP model had an average training speed of 0.6 seconds.

Figure 2: MLP confusion matrices

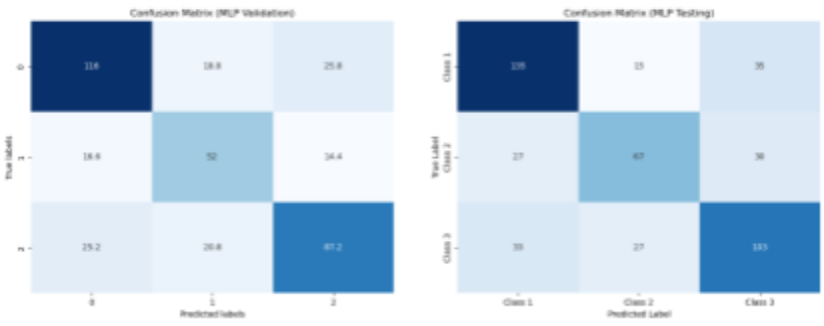
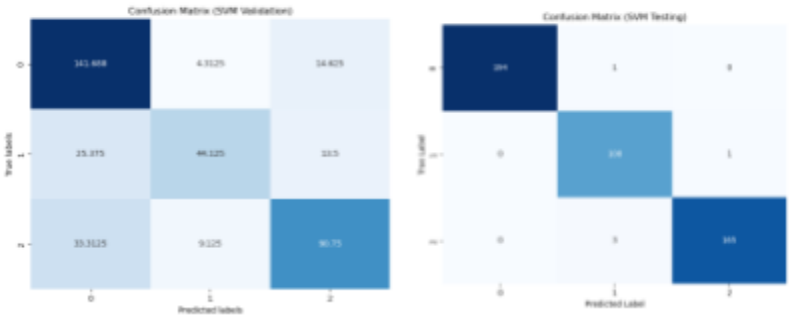


Figure 3: SVM confusion matrices



The SVM model also shows a higher rate of true positives and a lower rate of false positives than the MLP model on the validation and test set, as shown in the Receiving Operating Characteristic (ROC) curves in Figure 4 and Figure 5. In conclusion, SVM performed better than MLP on this dataset. This could be attributed to SVM's ability to handle high dimensional data better than MLP. Although nine features is not a considerably high number of features compared to datasets with up to hundreds or even thousands of features, it still is not a small number.

Figure 4: MLP ROC curves

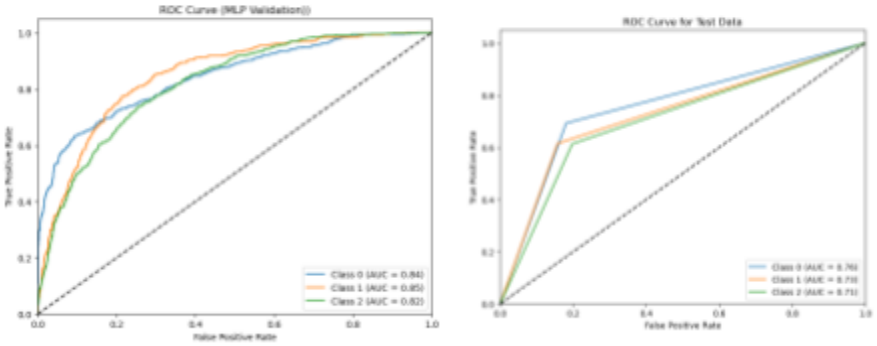
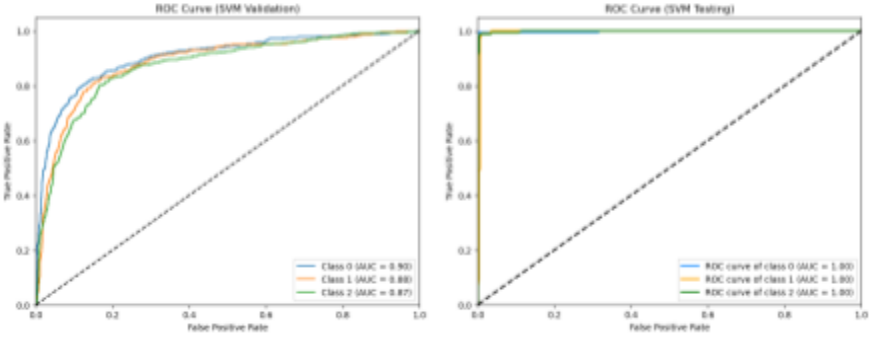


Figure 4: SVM ROC curves



SVM outperforms MLP when there are more features because MLPs can be affected by dimensionality. SVMs are also more robust to overfitting and to noise. In regards to overfitting, SVMs leverage maximising the margin between different classes, and they utilise kernel tricks to transform data into higher dimensional spaces to better capture nonlinear patterns. SVMs are more efficient in dealing with noise in data with kernel tricks, specifically the radial basis function which was used in the SVM model in this paper. Another reason for SVM's better performance on this dataset can be attributed to SVMs ability to handle smaller datasets. MLPs tend to need a lot of information, but this dataset is relatively small. Lastly, SVMs are more interpretable than MLPs. It is easier to understand the decision boundary in SVMs in regards to its parameters, but MLPs are often regarded as a "black box model" because they are more difficult to interpret. Because of this, the hyperparameter tuning was a longer and more in depth process. Even after finding the best hyperparameters through grid search, we cannot be certain as there are many more combinations of the changeable parameters such as activation functions, optimizers, number of hidden neurons, and hidden layers. It is difficult to pinpoint which hyperparameters needed more adjustments due to its lack of interpretability.

6. Lessons Learned and Future Work

From this study, several lessons can be learned regarding the performance of MLP and SVM models in classifying women's contraceptive methods based on socioeconomic factors. The entire process, including the data exploration, hyperparameter tuning, training, and testing, demonstrated the importance of understanding the nature of the data and choosing the appropriate models, architecture, and hyperparameter values. These factors are crucial to the effectiveness and quality of the models. When choosing between the two models, it is imperative to understand the data, the goal, and the nature of the models to choose the most

suitable one. Although SVM outperformed MLP in this study, MLP could be a better choice on a different task. MLP is a popular and powerful method, but it involves a lot of finetining as it is a “black box” model. Given more time and computational resources, the results of the MLP could be improved.

Many tasks can be explored for future work. The discrepancy between SVM’s performance on the validation and test set should be further investigated beyond what was done in this study given the time constraints. Another task would be to implement a more in-depth investigation on the features and their contributions to the models’ performances. In addition, other models beyond SVM and MLP can be explored, as well as exploring ensemble methods to produce even better performance results.

References

[1] T.-S. Lim, “UCI Machine Learning Repository,” *archive.ics.uci.edu*. https://archive.ics.uci.edu/dataset/30/contraceptive+method+choice (accessed Apr. 07, 2024).
[2] N. Hill, M. Siwatu, and M. Granger, “Safe Sex, Safe Communities: Analyzing the Link Between Contraceptive Usage and Crime Rates.” Available: https://swer.wtamu.edu/sites/default/files/Data/89-106-280-1044-1-PB.pdf
[3] S. Al-Stouhi and C. K. Reddy, “Transfer learning for class imbalance problems with inadequate data,” *Knowledge and Information Systems*, vol. 48, no. 1, pp. 201–228, Aug. 2015, doi: https://doi.org/10.1007/s10115-015-0870-3.
[4] M.-C. Popescu, V. Balas, L. Perescu-Popescu, and N. Mastorakis, “ Multilayer Perceptron and Neural Networks .”
[5] A. Rakhecha, “Understanding Learning Rate,” *Medium*, Jul. 07, 2019. https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de# (accessed Apr. 07, 2024).
[6] B. Ding, H. Qian, and J. Zhou, “Activation functions and their characteristics in deep neural networks,” *2018 Chinese Control And Decision Conference (CCDC)*, Jun. 2018, doi: https://doi.org/10.1109/ccdc.2018.8407425.
[7] Y. Jung and J. Hu, “K-fold averaging cross-validation procedure,” *Journal of Nonparametric Statistics*, vol. 27, no. 2, pp. 167–179, Feb. 2015, doi: https://doi.org/10.1080/10485252.2015.1010532.
[8] J. Shawe-Taylor and S. Sun, “Kernel Methods and Support Vector Machines,” in *Academic Press Library in Signal Processing: Volume 1 - Signal Processing Theory and Machine Learning*, 2014. doi: https://doi.org/10.1016/B978-0-12-396502-8.00016-4.
[9] G. Zhang, C. Wang, B. Xu, and R. Grosse, “Three Mechanisms of Weight Decay Regularization,” presented at the International Conference on Learning Representations,
[10] J. Brownlee, “A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks,” *Machine Learning Mastery*, Apr. 20, 2019. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, no. 16, pp. 321–357, Jun. 2002, doi: https://doi.org/10.1613/jair.953.
[12] J. Heaton, “The Number of Hidden Layers,” *Heaton Research*, Dec. 28, 2018. https://www.heatonresearch.com/2017/06/01/hidden-layers.html

Appendix 1 - Implementation Details

The implementation of a multilayer perceptron (MLP) and support vector machine (SVM) was done in Python. The first step involved data preprocessing. Because this task involved multiclass classification with three classes, the data was plotted in a bar graph to examine the distribution. In addition, the dataset only had 1473 rows.

After data preprocessing, I worked on the MLP model. I used the code from the group coursework as a reference. Building on this code, I adjusted it to fit the task and dataset at hand because I used two different datasets for the group coursework and the individual coursework. Before beginning the parameter tuning process, I split the training data into training sets and testing sets. Additionally, within the grid search, I used k-fold with five folds for cross validation. Initially, the parameter tuning stage was showing low accuracy at around 0.3 - 0.4. As this dataset had more features, I added more hidden layers to see if the model would perform better. Adding a single layer at a time, I added up to five layers. However, instead of improving, the accuracy rating was getting lower. At five layers, I decided to revert to two hidden layers as this yielded the best accuracy results, and the computational intensity and training time of adding this many layers were very high. Then, I added momentum and regularisation parameters. With this adjustment, the results slightly improved, but it was still not enough. I changed the values of the hidden neuron size and the learning rate that the grid search was iterating through. After doing this, I noticed that the model's performance was getting better as the hidden neuron size increased. In increments of 5, I adjusted the array of the hidden neuron size until the maximum accuracy rate was reached. Then, increased the number of epochs. The number of epochs was originally set to 200. Using increments of 100, I increased the epoch number until the maximum accuracy rate was reached, which was at 500 epochs. Then, I tried different activation functions. The original activation function was Rectified Linear Unit (ReLU). I tested Sigmoid, Tanh, and Exponential Linear Unit (ELU), ReLU showed the best results. Finally, I tried different optimizers. The original optimizer being used was Stochastic Gradient Descent (SGD). I tested Adam, Adagrad, and Adadelta. Of these, Adam showed the best results. Because Adam already uses momentum, I took out the momentum values in grid search. Then using these parameters found in the hyperparameter tuning phase, I created the model and trained it on the training data. Then, I tested the model on the testing data.

After the MLP model, I worked on the SVM model. Like the MLP model, I split the data into 80% training data and 20% testing data, and used k-fold validation with five folds for cross validation during grid search. In the grid search, I iterated through values for regularisation (C parameter), kernel type, and the kernel coefficient for Radial Basis Function (RBF). The kernel type RBF with C value of 1 and gamma value of 1 showed the best results. This already showed good results for the accuracy, but I wanted to test more parameters so I added values for class weight, shrinking heuristics, probability estimation, and kernel cache size. However, I found that only the class weight parameter made a difference in the accuracy results. After finding the best values for these parameters, I created the model and trained it on the training data. Then, I tested the model with the testing data.

Appendix 2 - Intermediate Results

The intermediate results for MLP showed low accuracy at around 0.3 - 0.4. I used the implementation details described in Appendix 1 to reach a higher accuracy result. However,

even after the hyperparameter tuning also described in Appendix 1, the results plateaued at 0.64 accuracy rate. Additionally, I followed the guidelines provided by J. Heaton to determine the number of hidden neurons, but these gave worse results. After experimenting, I found that the best number fell outside of Heaton's guidelines. Heaton's guidelines serve as a good start to begin experimenting, but MLPs are highly dependent on the architecture and data complexity. In the data exploration, the chi-square and p-values were calculated for feature selection. Only one feature, "wife working", showed values that indicated unlikely correlation. However, I tested results with and without this feature, and results worsened when this feature was not included. As a result, I included it in the model.

Appendix 3 - Glossary

1. Wife's age (numerical)
2. Wife's education (categorical) - (low) 1, 2, 3, 4 (high)
3. Husband's education (categorical) - (low) 1, 2, 3, 4 (high)
4. Number of children ever born (numerical)
5. Wife's religion (binary) - 0 (Non-Islam), 1 (Islam)
6. Wife's now working? (binary) - 0=Yes, 1=No
7. Husband's occupation (categorical) - 1, 2, 3, 4
8. Standard-of-living index (categorical) - (low) 1, 2, 3, 4 (high)
9. Media exposure (binary) - 0 (Good), 1 (Not good)
10. Contraceptive method used (class attribute) - 1 (No-use), 2 (Long-term), 3 (Short-term)
11. SMOTE - Synthetic Minority Oversampling Technique, a data rebalancing technique that uses oversamples the minority class by creating synthetic data (N. V. Chawla et al, 2002)
12. Learning rate - a hyperparameter in neural networks that specifies the adjustments of weights in gradient descent (A. Rakhecha, 2019)
13. Activation function - a function in a neural network that determines an output based on weights and input (B. Ding et al, 2018)
14. Feedforward - a type of neural network where the information or data moves in one direction
15. Regularisation - a method that adds penalty to the loss function to prevent overfitting
16. K-fold cross validation - a cross validation method that splits the dataset into k subsets and uses one for validation while k-1 subsets are used to train the model (Y. Jung and J. Hu, 2015)
17. Kernel - a function that transforms the data into a higher dimensional space in the context of a support vector machine to find a decision boundary (J. Shawe-Taylor and S. Sun, 2014)
18. Kernel coefficient - a parameter that determines the scale of the kernel's influence on the classification
19. Weight decay- also called L2 regularisation, a method to improve the generalisation of neural networks by penalising large weights during training (G. Zhang et al)
20. Receiving Operating Characteristic curve - an evaluation method of machine learning models that plots the rate of true positives and false positives
21. ReLU - rectified linear activation function, a type of activation function that directly outputs the input if it is positive, and outputs zero if it is not positive (J. Brownlee, 2019)