A comparison of Multilayer Perceptrons and Support Vector Machines in identifying people's income

Yousef Gharib

Abstract: This paper aims to investigate the performance of Multilayer perceptron (MLP) and Support Vector machines (SVM) in a binary classification task with an imbalanced dataset. SMOTE with random under-sampling is applied to the dataset to counter the data imbalance, its effects are evaluated a long with both models with optimal hyperparameters that are found through grid searches. The main metrics used in the evaluation are accuracy, specificity, and Receiver operation curves (ROC), it was found that SMOTE with random under-sampling does increase the overall performance of both models, however, MLP performed slightly better compared to SVM, as expected.

1. Brief description and motivation of the problem

The income of a population provides valuable insight on the economic status of a country allowing for estimations on prices of goods and quality of life. Income also validates its importance from an investment point of view in the form of equities, real estate, and currency trading/investment [1].

The aim of this paper is to analytically compare two machine learning models in a supervised binary classification task to predict if an individuals' income is greater than $50k. The models used in this paper are Multilayer Perceptron (MLP) and Support Vector Machines (SVM). Tests are carried out on each model by altering hyperparameter and optimization combinations to produce significant results and are critically evaluated to assess the performance of the selected models. The implementation of SMOTE and under-sampling is also investigated.

2. Dataset

The dataset used in this research was sourced from Kaggle and provides information on individuals such as age and education, and whether their income is over 50 thousand dollars, being the response variable. The dataset contains 43957 rows and 15 variables comprised of 6 continuous and 8 categorical columns. Approximately 76% of the response variable ***income_50K*** is classified as 0 (individuals' income is less than 50k) and 24% classified as 1.

Data preprocessing

Before model building and training, data preprocessing is done to ensure the data is easier to work with. Figure 1 shows histograms of continuous variables providing insight on the distribution of features, the variables are heavily skewed and scaling the data is necessary, a min-max scaler is applied to the data to transform the data to be between 0 and 1.

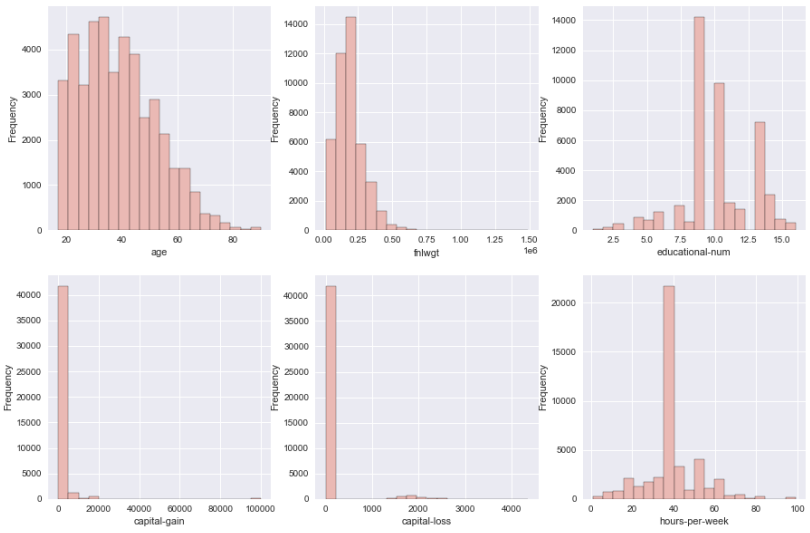

Figure 1: Histograms of continuous variables

The dataset contains 5767 missing values, removing the rows corresponding to these values leaves 40727 rows of data to work with. Outliers are found by calculating the Z-score and are removed if the score is outside the range [-3,3], leaving 37250 rows for the task. After removing null values and outliers, 76% of the data is still classified as 0 (majority) while 24% is classified as 1 (minority). The data is mainly comprised of categorical

features, which is problematic as neural networks only work with numeric data and removing them will result in significant loss of data, therefore the categorical columns are transformed using one hot encoding, the resulting dataset contains 37250 rows and 101 columns excluding the response variable.

The dataset is split into train/test subsets made up of 75%/25% of the data respectively, once this is done the issue of imbalanced data can be addressed as we do not want synthetic data present in the test set, the train/test split sizes may be further adjusted to significantly reduce model training times. SMOTE with random under-sampling strategy of 0.5 on the majority class is applied to the training data to reverse the bias [2] on the majority class as well as balance the class distribution in the training data.

3. Summary of the two neural network models with their pros and cons

Multilayer Perceptron (MLP)

Multilayer perceptron (MLP) is a branch of feed-forward neural network and excels in supervised classification and regression tasks as well as pattern recognition [3]. The structure of a MLP consists of an input layer, hidden layer(s), and an output layer where each layer contains neurons and are usually fully connected to each other through the neurons. Initially, weights and biases are assigned to each connection between layers and an activation function is applied to each neuron to activate or deactivate it. The weights and biases are tuned and updated during learning through backpropagation.

MLP is widely used especially in image processing because of its ability to learn on a given dataset, it can be easily adjusted for more complex data by increasing the number of hidden layers or neurons. In addition to that, MLP can be very useful when complex mathematical models cannot be composed [4].

However, MLP is also described as a black box as it is unclear how feature importance is determined. Another disadvantage of MLP is the number of parameters that need to be optimized, a simple fully connected network can have thousands of adjustable parameters, this may result in inefficiency if too many nodes or hidden layers are used, also making it more prone to overfitting. Since MLP only works with numeric data, more preprocessing must be done with categorical data converting it to numeric data.

Support Vector Machines (SVM)

Support vector machines (SVM) are a robust learning algorithm used in classification and regression tasks. SVM aims to construct a hyper-plane that separates classes, the hyper-plane constructed maximizes the distance of the nearest data point for each of the classes [5]. Data is usually not linearly separable, in this case, a kernel is applied to transform the data allowing for the data to be linearly separable.

SVM provide good out-of-sample generalization making it less likely to overfit and potentially ignoring some bias in training sets [6]. SVM is also highly effective in cases with high dimensionality data, particularly when the number of dimensions is greater than the number of samples. Applying a kernel to the data allows the SVM to be very flexible when optimizing margin distance.

Since SVM only works in binary classification tasks, the data must be transformed to allow for a “1vs1” or “1vsAll” situation among classes, transforming multiclass tasks into binary classification problems. SVM struggles with very noisy data containing overlapping classes and is not suitable for very large datasets since the computation time scales exponentially with datapoints as it works with quadratic equations. Generally, SVM do not provide class probabilities since a decision boundary is used, however, can be loosely calculated through an expensive 5-fold cross-validation as used in this paper [7].

4. Hypothesis statement

Both SVM and MLP are reliable models and no major difference in performance is expected between them, however, it is expected that MLP will perform slightly better because of its ability to learn and its versatility through altering all its hyperparameters.

Implementing SMOTE is expected to increase the specificity and reduce the accuracy of both models, since SMOTE and under-sampling reverses the bias of the majority class and the test set is stratified, in addition to this, SVM is expected to perform worse compared to MLP as implementing SMOTE interpolates data points that may result in classes overlapping more, significantly increasing difficulty to construct the optimal hyper-plane. Since SVM is computationally expensive and SMOTE is implemented, it is expected to take significantly longer to train the model compared to MLP.

The complexity of a dataset plays a major role in how many hidden layers and neurons result in the best performance; it is expected increasing the number of hidden layers in the network will have negative effects on the model as the dataset is not complex.

5. Choice of training and evaluation methodology

The data is separated into a 75%/25% train/test split and with a stratified test set to maintain the imbalance in the original dataset. The optimal hyperparameters are found through a grid search with 5-fold cross validation for each combination of hyperparameters to avoid overfitting. SVM grid search is trained on a 40%/60% split to reduce time taken, using a larger dataset is not expected to affect the results of the grid search. Investigation of implementation without SMOTE and varying degrees of SMOTE and random under-sampling also carried out and compared for each model.

Accuracy and specificity are the main evaluation metrics used as while both classes are important, specificity allows us to investigate predictions of the minority class. Other metrics such as F1 score, sensitivity are also assessed to further investigate the performance of each mode. Finally, ROC curves are used to investigate the compromise between sensitivity and specificity.

6. Choice of parameters and experimental results

Multilayer Perceptron

The initial structure used in hyperparameter optimization for MLP involved 1 hidden layer with twice the number of neurons as in the input layer, dropout is used to reduce chances of overfitting and a SoftMax function is applied to ensure the final outputs sum to 1. Grid search with 5-fold cross validation is used to this structure to find the optimal configuration for the hyperparameters. Activation function is the first hyper parameter tested, it was found that RELU outperformed sigmoid and is used in further testing of other hyperparameters. The next hyperparameters tested were learning rate, optimizer, and loss criterion, table 1 displays the results of this grid search. The number of hidden layers and neurons used are the final hyperparameters tested, it was found the increasing the number of hidden layers decreases the accuracy while around 400 neurons resulted in the highest accuracy (results shown in Appendix 2). The MLP grid searches were carried out with a maximum of 300 epochs and early stopping criteria as most training did not improve significantly by the 300^{th} epoch.

In addition to this, momentum was not tested as it only affects SGD and Adam optimizer provided the best results.

The final model was trained with 1 hidden layer, 400 hidden neurons, Adam optimizer, negative log likelihood loss with a learning rate of 0.00001.

Support Vector Machines

To keep the experiment fair, SVM hyperparameter optimization was also done using a grid search with 5-fold cross-validation. Hyperparameters optimized on SVM are a regularization parameter C, gamma coefficient and kernel function, table 2 displays the results of the grid search, another hyperparameter that could be tested with is degree, however, this only affects the polynomial kernel function, excluding degree from the grid search reduces the number of combinations significantly, with 4 polynomial degrees the number of parameter combinations increases from 64 to 256 and was therefore removed to decrease grid search time.

Optimizer	Learning Rate	Loss criterion	Mean score	Standard deviation score	C	Gamma	Kernel	Mean score	Standard deviation score
Adam	0.00001	CrossEntropyLoss	0.786476	0.009788	10	0.001	linear	0.777492	0.028684
Adam	0.01	CrossEntropyLoss	0.500000	0.000061	10	0.001	rbf	0.770872	0.028073
Adam	0.001	CrossEntropyLoss	0.499961	0.000047	10	0.001	poly	0.548164	0.096570
Adam	0.00001	NLLLoss	0.797153	0.007315	10	0.001	sigmoid	0.756232	0.033229
Adam	0.0001	CrossEntropyLoss	0.786709	0.008049	100	1	linear	0.777444	0.028617
Adam	0.1	CrossEntropyLoss	0.500000	0.000061	100	1	rbf	0.843401	0.030308
Adam	0.01	NLLLoss	0.500000	0.000061	100	1	poly	0.815521	0.028360
Adam	0.001	NLLLoss	0.557948	0.099232	100	1	sigmoid	0.437765	0.012557
Adam	0.0001	NLLLoss	0.794678	0.003675	100	0.1	linear	0.777444	0.028617
Adam	0.1	NLLLoss	0.500000	0.000061	100	0.1	rbf	0.831708	0.019228
SGD	0.0001	CrossEntropyLoss	0.657125	0.007393	100	0.1	poly	0.829195	0.013022
SGD	0.00001	CrossEntropyLoss	0.432657	0.026977	100	0.1	sigmoid	0.656212	0.040611
SGD	0.00001	NLLLoss	0.432928	0.027111	100	0.001	linear	0.777444	0.028617
SGD	0.1	NLLLoss	0.612603	0.008551	100	0.001	rbf	0.820981	0.006313
SGD	0.001	CrossEntropyLoss	0.751701	0.016873	100	0.001	poly	0.774739	0.018539
SGD	0.01	NLLLoss	0.500116	0.000061	100	0.001	sigmoid	0.755700	0.037020
SGD	0.001	NLLLoss	0.764892	0.006391	100	0.0001	linear	0.777444	0.028617
SGD	0.01	CrossEntropyLoss	0.737429	0.003848	100	0.0001	rbf	0.779038	0.028042
SGD	0.0001	NLLLoss	0.717351	0.024380	100	0.0001	poly	0.548164	0.096570
SGD	0.1	CrossEntropyLoss	0.677549	0.006646	100	0.0001	sigmoid	0.776381	0.029130

Table 1: Results of grid search for MLP and SVM

7. Analysis and critical evaluation of results

Result comparison

As expected, MLP performed slightly better compared to SVM as shown in figure 2, while this was expected, the grid search results shown in the tables suggest SVM had a higher mean validation score, this may be due to changes made after further testing with SMOTE and dataset sizes, the grid search was carried out with 40% test size on SVM to significantly reduce time taken, and 25% test size for MLP, while this may not be completely fair, the grid search is carried out to find optimal hyperparameters, and is not expected to change

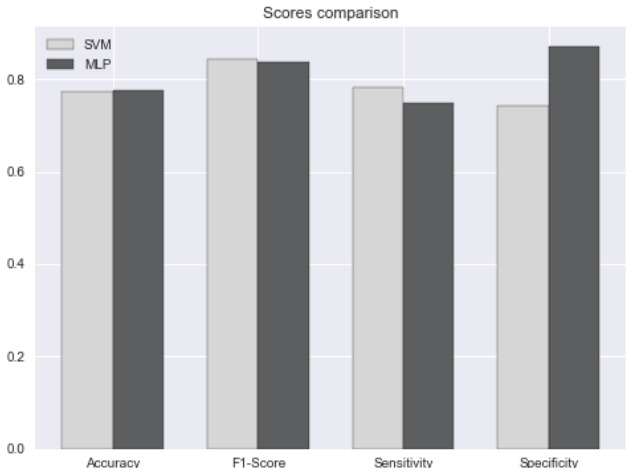

Figure 2: Comparison of metric scores for both models

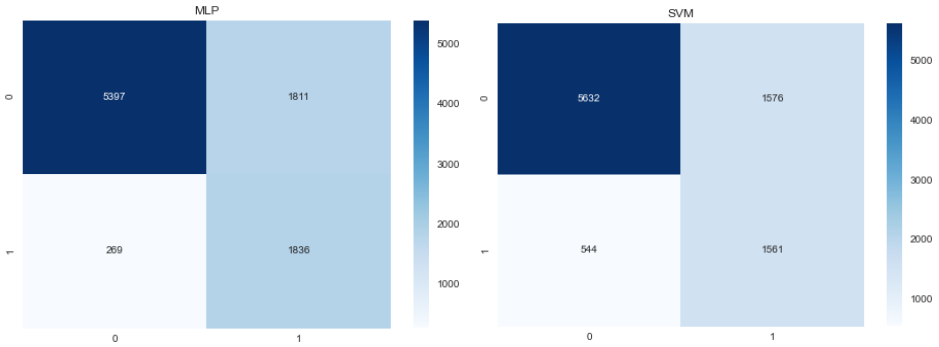

Figure 3: Confusion matrix for both models

based on the training set size, the final models are trained on the same size dataset. The MLP also has a surprisingly high specificity of approximately 87%, which is a very high score considering SMOTE was used to balance the classes, although MLP had higher scores for accuracy and specificity, SVM had higher F1- score and sensitivity, suggesting it predicts the majority class more accurately. The confusion matrix shown in figure 3 shows both models had a very similar distribution of classifications, however, SVM has double the number of false negatives compared to MLP. The ROC curve for both models were plotted to display the overall performance of each classifier, it is clear from figure 4 that MLP performed significantly better compared to SVM with an area under the curve of approximately 0.89 while SVM had an area of around 0.76. Overall, MLP outperformed SVM, but SVM is a better predictor of the majority class while MLP is a better predictor of the minority class.

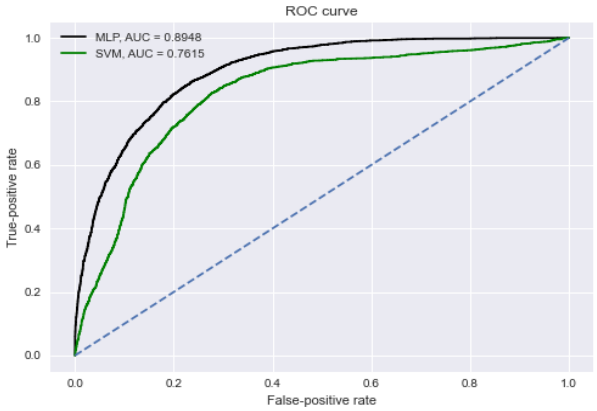

Figure 4: ROC curve for both models

Time comparison

Many steps were taken to significantly reduce the time taken for both grid searches, such as excluding certain hyperparameters and training using smaller sizes, for a fair comparison, both models were trained using a 75%/25% train/test split. Other train/test ratios were manually tested, and it was found 75%/25% provided the best outcome for both models. The time taken in the grid search and training of final model for MLP was significantly lower compared to SVM even after taking steps to reduce the time for SVM. It took approximately 2 hours for the grid search of MLP and over 6 minutes for training, while SVM took over 3 hours for the grid search and just under 3 minutes for training, this is surprising as SVM was expected to take longer to train than MLP, this possibly suggests the SVM hyperparameters very good and allow for the optimal hyperplane to be easily found.

Sampling method comparison

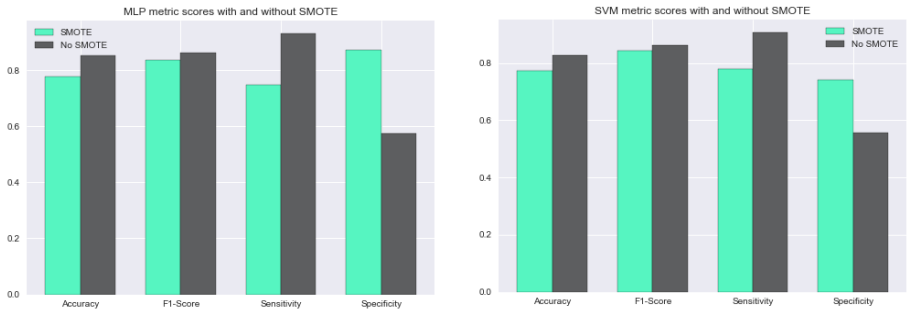

Figure 5: Comparison of metrics on models with and without SMOTE

The effects of SMOTE and under sampling were investigating by training the models without SMOTE and different under sampling strategy ratios and were compared, the effects of the investigation were the same for both models. When SMOTE was not used, the accuracy of both models was significantly higher, approximately 85% compared to 77% with SMOTE, however, the specificity of both models is around 50%, suggesting the models badly predict the minority class, the accuracy without SMOTE is high likely because of the imbalance of classes in the training set, meaning the models would not perform well on balanced test data.

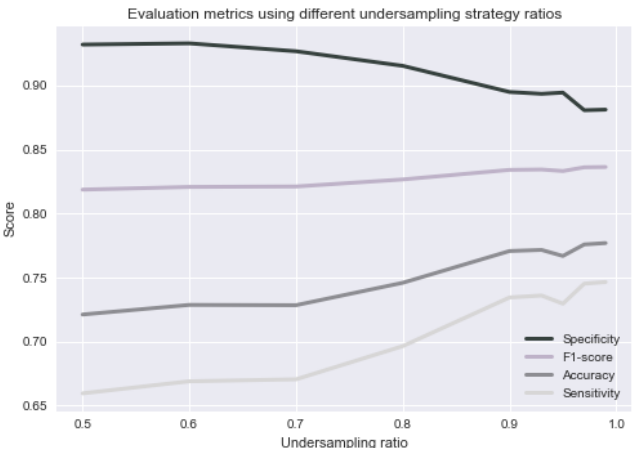

Figure 6: Metric scores based on under sampling ratio for MLP

Varying the under-sampling ratio had significant effects on the metric scores for both models, it was found that using a score of 0.99 provided the best overall scores. Figure 6 shows that when the ratio was increased, all metrics except specificity had increasing scores with a higher under sampling ratio.

8. Conclusions

The aim of the paper was to compare MLP and SVM on a binary classification task for an unbalanced dataset. While both models had similar results, MLP proved to be a more reliable and accurate model through the ROC curve and other evaluation metrics, as hypothesized. Both models were also compared with and without the implementation of SMOTE and under-sampling and it was found that using decreased the accuracy but significantly increased the specificity, it is expected the accuracy decreased because of the imbalance of classes in the test set and that in a balanced set it would increase.

Lessons learned and future work

Finding the optimal hyperparameters for MLP can be tedious and exhausting compared to SVM since there are more hyperparameters to tune resulting in a significantly higher number of possible combinations. SMOTE and under-sampling can play a major role in the performance of a model as shown in the comparison, testing different combinations of SMOTE and under-sampling is necessary for an imbalanced dataset.

Future work may involve varying the order of hyperparameter optimization carried out for MLP to see if any significant difference in performance is present. In addition to that, investigating the performance of boosting as well as ensemble methods on both models may provide interesting insight on increasing performance.

References

[1] Western Asset (2012). The Importance of Income.
[2] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp.321–357
[3] Why MultiLayer Perceptron/Neural Network? (n.d.)
[4] Gardner, M.W. and Dorling, S.R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment, [online] 32(14-15), pp.2627–2636
[5] V.Kecman (2013). Support Vector Machines: Theory and Applications.
[6] Auria, L. and Moro, R. (n.d.). A Service of zbw Leibniz-Informationszentrum Wirtschaft Leibniz Information Centre for Economics.
[7] https://scikit-learn.org/stable/modules/svm.html
[8] Freund, Y. and Schapire, R.E. (1999). A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence, [online] 14(5), pp.771–780.
[9] Schwenk, H. and Bengio, Y. (2000). Boosting Neural Networks. Neural Computation, 12(8), pp.1869–1887.

Appendix 1 - Glossary

Terminology	Definition
Hyperparameter	A parameter which affects and controls machine learning models in the learning process.
Activation function	A function applied to the input of a node that switches the node on or off.
Loss criterion	A function used to evaluate the error from adjusting the weights and biases in a neural network.
Optimizer	An algorithm used to adjust the weights and biases of a network by finding the minima of a function.
Learning rate	A step size applied to updated weights and biases controlling how quickly a model trains as well as avoiding overfitting.
Hyperplane	A plane in a space with dimension 1 less than the number of features, used as a decision boundary in classification tasks.
Min-max scaler	A transformation applied to a dataset to translate data points to a range between 0 and 1.
Z-score	A measure used to investigate the relationship between a datapoint and the mean and standard deviation of all the points, allowing for easier identification of outliers.
One hot encoding	A transformation applied to categorical variables, creating a new binary column of each category, and assigning a value of 0 if false and 1 if true.
SMOTE	An oversampling technique used to generate datapoints for the minority class of an imbalanced dataset, SMOTE stands for Synthetic Minority Oversampling Technique.
Linearly separable	A mathematical term used to represent if two classes or sets can be separated with a hyperplane.
Kernel function	A function applied to a dataset to manipulate the data to help find an optimal hyperplane.
Cross validation	A resampling technique that splits data into folds, training on one fold and testing on the others. This is done to each fold and reduces overfitting.
Momentum	A technique applied to an optimization method to assist the optimizer in finding the minima of a function more efficiently, this only applies to some optimizers such as stochastic gradient descent.
C parameter	A hyperparameter of SVM deciding how much misclassification to avoid, used to control error.
Gamma coefficient	A hyperparameter influencing the curvature of the hyperplane decision boundary.
Accuracy	A metric used to evaluate a model, calculated by finding the ratio of correct predictions to overall predictions.
Sensitivity	A metric used to evaluate how well the model predicts true cases. Calculated by dividing the number of correctly predicted true cases by the total number of cases predicted as true, also known as precision
Specificity	A metric used to evaluate how well the model predicts false cases. Calculated by dividing the number of correctly predicted false cases by the total number of cases predicted as false.
F1- score	A measure of accuracy when a class is considered more important than the other. Calculated from the sensitivity and recall.
ROC curve	A graph that displays the ability of a model to classify with varying thresholds, where the area under the curve summarizes the ROC curve for all thresholds.

Appendix 2 - Implementation details

This evaluation was carried out using python, three jupyter notebook files are used to keep the code organized and easy to use. The jupyter notebooks are organised as such:

- Data pre-processing and MLP hyperparameter optimization
- SVM hyperparameter optimization
- Training of best models for both MLP and SVM as well as testing

Intermediate Results

Below are tables of results of evaluation metrics based on altering the number of hidden layers and the number of neurons when 1 hidden layer is used. We see that 1 hidden layer with 400 neurons provides the best scores.

Number of hidden layers	Accuracy	F1-score	Sensitivity	Specificity
1	0.7267	0.7907	0.6670	0.9311
2	0.7185	0.7826	0.6546	0.9372
3	0.7063	0.7757	0.6367	0.9444
4	0.6964	0.7604	0.6223	0.9501
5	0.6836	0.7476	0.6054	0.9515

Number of neurons	Accuracy	F1-score	Sensitivity	Specificity
100	0.7215	0.7858	0.6599	0.9325
200	0.7220	0.7862	0.6605	0.9325
300	0.7196	0.7838	0.6569	0.9344
400	0.7228	0.7868	0.6607	0.9353
500	0.7143	0.7781	0.6473	0.9439

It is expected that the number of hidden layers would not make significant improvements since the data is not extremely complicated, however it was not expected to reduce accuracy and other metrics.

Below is the ROC curve comparing the model with and without the implementation of SMOTE, the area under the curve is slightly higher without using SMOTE, however implementing SMOTE significantly increased other metrics.