A Comparative Analysis of Multilayer Perceptrons and Support Vector Machines in Predicting Credit Card Payment Defaults.

Deenadhayalan Ravi

deenadhayalan.ravi@city.ac.uk

Abstract

The objective of this study is to assess the effectiveness of Multilayer Perceptron (MLP) and Support Vector Machines (SVM) algorithms in predicting credit card payment defaults. The performance of both models is evaluated by tuning their hyperparameters using a grid search method and validated using a stratified cross-validation approach. The results obtained from the best performing models are compared using Confusion Matrix, Receiver Operating Characteristic Curve (ROC Curve) and the classification report.

1. Introduction

In this modern economy, nearly one fourth of the world's population is having credit card and it has become a ubiquitous and widely accepted form of payment. Furthermore, due to the widespread adoption of credit cards, the risk of payment default has become a major concern for financial institutions, highlighting the need for accurate predictive models to manage risk and prevent potential losses.

This paper aims to provide a critical evaluation of the effectiveness of two models, Multilayer Perceptron (MLP) and Support Vector Machines (SVM), in detecting credit card payment defaults. The study investigates the performance of these models under various configurations and data distributions, emphasizing the importance of developing accurate predictive models to manage the risk of credit card payment defaults. Additionally, the study provides insights into the suitability of MLP and SVM algorithms for tackling the problem of credit card payment defaults.

Section 2 of this paper provides a concise overview of the dataset utilized, data cleaning and descriptive analysis. Section 3 presents a detailed comparison of the approaches and methods employed during the implementation stage. In Section 4, we evaluate and critically compare the results of the implemented models. Finally, Section 5 concludes the paper, summarizing the findings and highlighting the significance of developing accurate predictive models for managing credit card payment defaults.

1.1. Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) is a type of Artificial Neural Network (ANN), which is commonly used for supervised learning tasks such as classification and regression. It consists of neurons connected in a feedforward manner from input layer to hidden layers and then to output layer. The model is initialized with random weights and biases.

The input data is fed into the input layer of the network, and the output is obtained by passing the data through multiple hidden layers with each layer applying an activation function to the weighted sum of its inputs. The loss function is then calculated by comparing the predicted output to the actual output. The error is propagated backwards through the network using backpropagation, and the weights and biases of the network are updated using an optimization algorithm such as Stochastic Gradient Descent (SGD) / Adam. MLP can learn complex non-linear relationships between input and output data. However, the performance of MLP depends on the quality and quantity of the training data, choice of hyperparameters, and optimization algorithm used.

1.2. Support Vector Machines (SVM)

Support Vector Machines (SVMs) are a popular supervised learning algorithm used for classification, regression, and outlier detection. SVMs work by finding the best boundary or hyperplane that can separate the data into different groups. The hyperplane is chosen to maximize the margin or distance between the hyperplane and the closest data points from each group. SVM can handle non-linearly separable data using a kernel function that transforms the data into a higher-dimensional space. SVMs have several advantages, including their ability to handle high-dimensional data and their effectiveness with small and medium-sized datasets. However, they can be sensitive to the choice of kernel function and hyperparameters, and they can be computationally expensive for large datasets.

2. Dataset

The dataset used in this analysis is 'default of credit card clients' from UCI Machine Learning Repository [1]. This dataset contains 30,000 instances with 23 independent attributes and 1 dependent attribute 'DEFAULT PAYMENT NEXT MONTH' with the class 0 as non-defaulters and class 1 as defaulters. In those 24 attributes, there are 9 categorical attributes like SEX, EDUCATION, MARRIAGE and Pay (PAY means on which month the customer has paid the bills) and the remaining attributes are continuous values like amount of month bill (BILL_AMT1 to 6) and amount of previous payments (PAY_AMT1 to 6) for 6 months.

By seeing the fig1 it is evident that the distribution of data on the target attribute it is evident that there is a class imbalance of 80/20 percentage of majority and minority class which is non defaulters and defaulters respectively. This is a challenge can be overcome by either going for under sampling or oversampling. The under sampling may lead to loss of information about the majority class. So, for the main comparison of MLP and SVM Synthetic Minority Oversampling Technique (SMOTE) is used in this study.

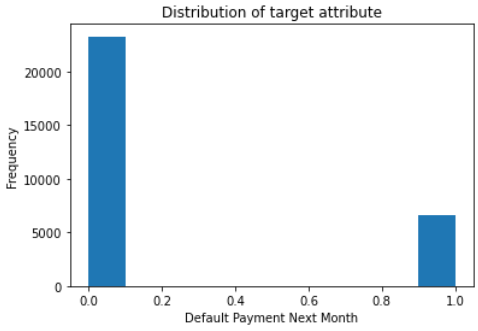

Fig1.Distribution of target attribute

2.1. Data preparation & Initial Descriptive Analysis

- The first step is to clean the data for null values, duplicates, dropping some unwanted columns, etc.
- The next step is to prepare the data for analysis, like applying label encoding or one hot encoding for the categorical attributes like SEX, MARRIAGE, EDUCATION and PAY_0 to PAY_6, both the methods are trailed on a baseline MLP.
- Then in order to overcome the class imbalance, initially down sampling is done and later found that SMOTE is performing better, the results of analysis of all four combinations like down sampling/SMOTE and label encoding/one hot encoding on a baseline MLP is shown in the appendix, where SMOTE with One Hot encoding works better. In label encoding all the classes of an attribute are labeled and kept in the same column, so the model sees that as a continuous number (say 0 to 5 for 5 class) where as

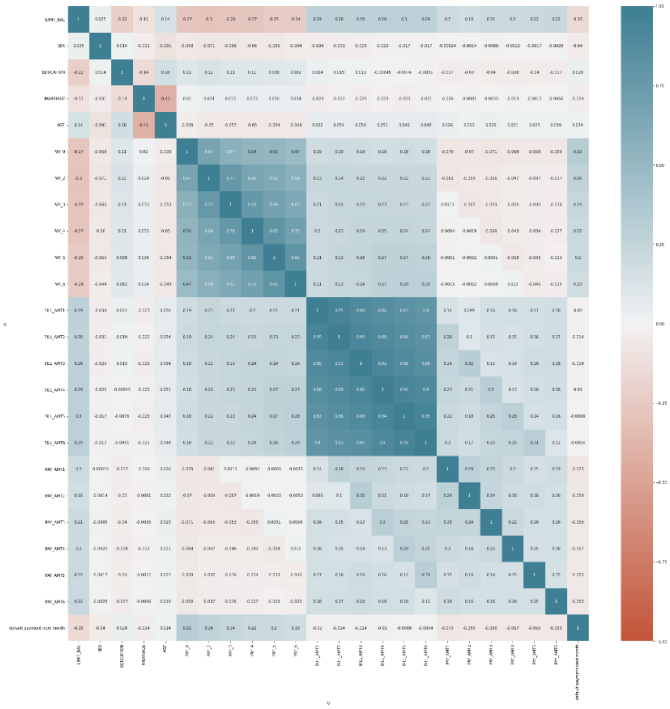

Fig2. Correlation Matrix

in OHE each and every class of an attribute will be in a separate column means in a separate neuron, so the model can learn better in OHE.

- Splitting the dataset into x - independent variables and y – dependent variable, and splitting train, validation and test dataset. Scaling the x_train and x_test sets for ease of computation and also to prevent the optimization from getting stuck in local optima.
- Created a correlation matrix to check the correlation of the target variable with the other variables and found that Limit balance (LIMT_BAL) and the history of past payment (pay_0 to Pay_6) are having a correlation of 0.2 each. And also, its evident that there is high multi collinearity on the variables BILL_AMT1 to BILL_AMT6 followed by PAY_0 to PAY_6 which can lead to can lead to unstable or unreliable regression coefficients, and may make it difficult to interpret the effects of each variable on the dependent variable.

3. Methods

In this section, we describe the methodology used for analysing the problem using MLP and SVM models, as well as the architecture and hyperparameters used in both the models.

3.1. Methodology:

i. **Train/Test Split:** In this analysis, the dataset is split into 75% and 25% for train and test respectively. 25% in the training is allocated for validation in the MLP Baseline Model which is written using Pytorch so grid search cross validation is not there, so, manual validated. In the main hyperparameter optimized MLP and SVM models using sklearn classifiers with grid search and cross validation.

ii. **Model Selection:** The subject dataset is in supervised tabular format so MLP and SVM is good choice to compare, former one is an artificial neural network and the later one is supervised classifier model. Using the GridSearchCV from sklearn.model_selection package is used to choose the best model with best parameters.

iii. **Model Training:** The model will be trained using suitable algorithms, for MLP_baseline using PyTorch a manual architectural code, used MLPClassifier() from sklearn.neural_network for MLP and used SVC() from sklearn.neural_network for SVM.

iv. **Hyperparameter Tuning:** using the parameter grid and GridSearchCV, the model can be trained in all the possible combinations of parameters and stratified cross validates with 5 fold and provides accuracy score against each combination. And the best trained model's parameter is pulled from the grid.

v. **Model Evaluation:** Evaluate the performance of the trained model by testing it the test data, and save the performance metrics such as accuracy, precision, recall, f1 score, ROC curve, confusion matrix. Save the evaluation results of the both the best trained model and compare.

3.2. Architecture and Parameters used for the MLP

The architectural parameters in hyperparameters are, number of hidden layers – adding more layers allows the model to learn complex relationships, but too many layers can lead to overfitting, number of neurons in the hidden layer - can increase the model's ability to capture complex patterns, but again, too many neurons can lead to overfitting, type of activation function of each layer - have a significant impact on the model's performance. One thumb rule for choosing the neuron size is that it should be 2/3 of the input neuron size.

The learning parameters in hyperparameters are, type of optimizers - determines how the weights are updated during training, learning rate - the step size used to update the weights during training. A high learning rate can cause the model to converge too quickly and potentially overshoot the optimal solution, while a low learning rate can cause the model to converge slowly or get stuck in a local minimum, momentum - speeding up the convergence

of the optimization process and preventing the network from getting stuck in a local minimum, type of loss functions, type of model evaluation metrics, epoch, early stop.

Lambda and dropouts are the two regularization parameters, this is a penalty term added to the loss function to prevent overfitting.

3.3. Architecture and Parameters used for the SVM

The architecture of an SVM is based on the concept of finding the optimal hyperplane that separates the data into different classes . The hyperparameters for SVM are kernel function, degree parameter, regularization parameter, gamma parameter and class weights. One advantage of SVMs over other machine learning algorithms is that they do not suffer from the problem of getting stuck in random local minima due to the fact that they do not start with random weights.

4. Results, Findings & Evaluation

4.1. Model Selection

Table 2 shows the top 10 combinations of the hyperparameters optimization grid search, where the left extreme is the MLP – architectural optimization and the center one is the MLP – learning and regularization parametric optimization and the right one is SVM combinations. The accuracy score mentioned in the table is mean of the 5-fold stratified cross validation. The model selection is based on this accuracy where our MLP baseline model created in PyTorch (Model1) has given 61%(Model1-val_acc@500th epoch) and using the hyper parameters from the below table with green highlighted row MLP gave 86% (Model3-CV) and SVM gave only 70% (Model6-CV).

Top 10 results of MLP Hyperparmeter Optimization with architectural parameters Grid search (Model3)				
mean_fit_time	param_activation	param_hidden_layer_sizes	mean_test_score	rank_test_score
16.88	tanh	70	0.87	1
15.29	tanh	120	0.87	2
11.50	logistic	(50,)	0.86	3
14.51	logistic	(20,)	0.86	4
11.93	relu	70	0.86	5
12.09	logistic	60	0.86	6
10.37	logistic	(20, 20)	0.86	7
19.45	logistic	120	0.86	8
10.38	tanh	(20,)	0.86	9
13.23	logistic	70	0.86	10

Top 10 results of MPL Hyperparmeter Optimization for learning and regularization parameter Grid search (Model3.1)							
mean_fit_time	param_alpha	param_learning_rate	param_learning_rate_init	param_momentum	param_solver	mean_test_score	rank_test_score
160.42	0.001	constant	0.01	0.5	sgd	0.86	1
162.16	0.0001	constant	0.01	0.5	sgd	0.86	2
149.93	0.0001	constant	0.01	0.2	sgd	0.86	3
145.99	0.01	constant	0.01	0.2	sgd	0.86	4
143.53	0.001	constant	0.01	0.2	sgd	0.86	5
144.04	0.0001	constant	0.01	0.1	sgd	0.86	6
143.25	0.001	constant	0.01	0.1	sgd	0.86	7
171.59	0.0001	adaptive	0.005	0.3	sgd	0.86	8
175.86	0.001	adaptive	0.005	0.3	sgd	0.86	9
168.93	0.01	adaptive	0.005	0.3	sgd	0.86	10

Top 10 results of SVM Hyperparmeter Optimization Grid search (Model6)						
mean_fit_time	param_C	param_degree	param_gamma	param_kernel	mean_test_score	rank_test_score
196.24	0.001	4	0.1	rbf	0.70	1
82.79	0.001	3	0.1	rbf	0.70	1
93.64	0.1	4	0.1	rbf	0.69	3
79.39	0.1	3	0.1	rbf	0.69	3
76.81	0.01	4	0.1	rbf	0.69	5
116.64	0.01	3	0.1	rbf	0.69	5
65.39	1	3	0.1	rbf	0.69	7
94.63	1	4	0.1	rbf	0.69	7
77.28	0.1	3	0.05	rbf	0.68	9
88.83	0.1	4	0.05	rbf	0.68	9

Table 1 - Hyperparameters Grid Search

4.2. Algorithm Comparison

The models with best hyperparameters of all the combinations of MLP and SVM (i.e. Model4 and Model7 resp.) are now tested on the test dataset which are unknown to the trained model. In this section we are going to compare the performance metrics like classification report, ROC curve, confusion matrix of both the final models.

First we started the analysis with the MLP baseline model created on pyTorch (Model1) where the overall accuracy on the test dataset is 60% and with MLP hyperparameter optimization the accuracy has risen to 81%(Model4), where as in SVM with hyperparameter optimization the accuracy is 70%(model7). All the three model's test accuracies are in line with their validation accuracies which means that the models have not overfitted, it is performing the same in the unknown test dataset.

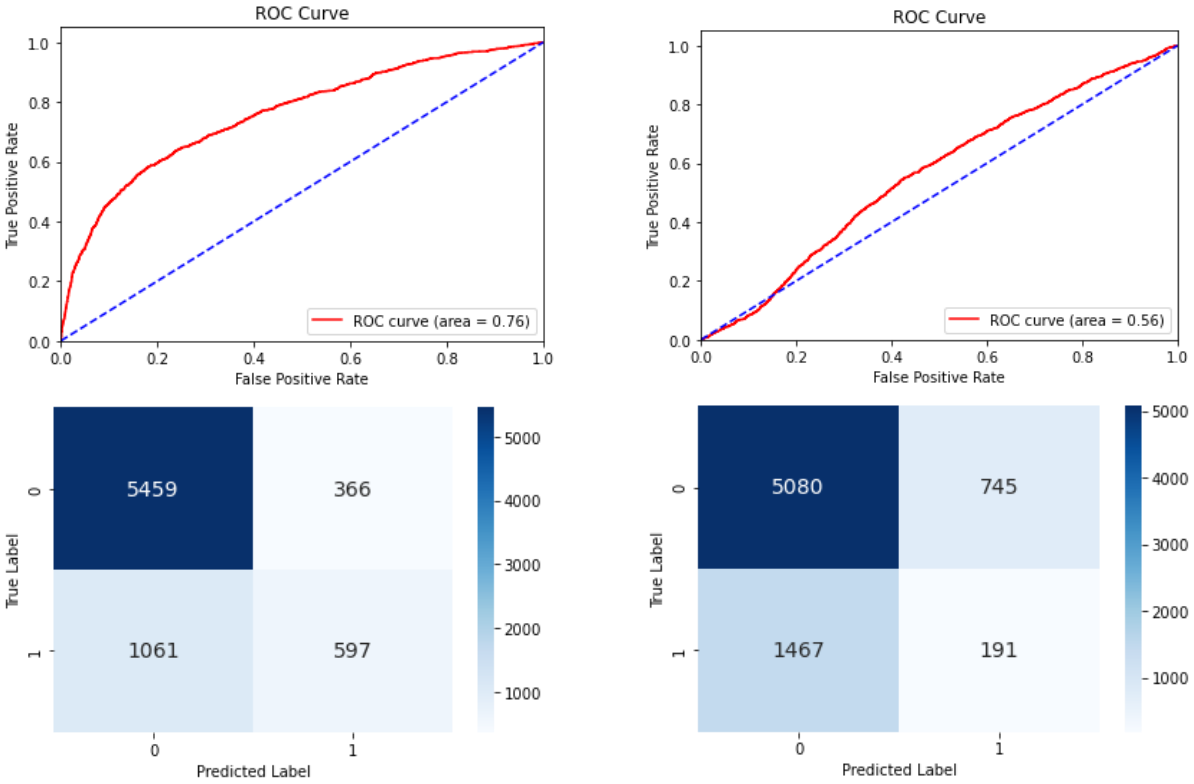

Fig3. ROC & Confusion Matrix of MLP (left) and SVM (right)

The ROC curve is plotted to check the quality of the classifiers and visually enrich the analysis. The area under the curve (AUC) which measures the accuracy of the classification is 0.76 MLP model, where as in SVM it is 0.56, which is much influenced by the minority class imbalance.

Since there is a class imbalance in the target attribute, relying only on the accuracy of the model is not sufficient, because we will get a good accuracy in predicting the majority class that is non-defaulters. Our target is to predict the defaulters that is class 1 (i.e. True Negatives), in technical terms the model should be trained well to have higher Specifity=TN/(TN+FP) and Negative Predictive Value(NPV) = TN / (TN+FN). So, we will even compare the classification report of both the models. Where NPV and Sensitivity of all the models have been highlighted in yellow in the classification report table2. In which MLP models have better NPV, Sensitivity and f1-scores than the SVM model, so MLP can predict the defaulters better than the SVM model.

Classification report all the models						
Model No.	Model	class	Precision / NPV	recall / Sensitivity	f1-score	accuracy score
1	MLP_baseline_ pytorch	0	0.82	0.64	0.71	0.60
		1	0.28	0.49	0.35	
4	MLP_Final_ Model	0	0.84	0.94	0.88	0.81
		1	0.62	0.36	0.46	
7	SVM_final_ model	0	0.78	0.87	0.82	0.70
		1	0.2	0.12	0.15	

Table2. Classification report of all the models

From the confusion matrix, the True Positives are 5459-MLP and 5080-SVM which is higher when compared to the other parameters like FN, FP, TN, which means having good precision and recall and that shows the model got trained better in the majority class. The True Negatives are 597-MLP and 191-SVM, because of minority class imbalance issue. Since the FP are higher in both the models compared to the FN, this is a type 1 error, need to find ways to lower this by tunning the hyperparameters.

5. Conclusion

Our study reviews how accurately two trained models, a Multilayer Perceptron and a Support Vector Machine can predict the defaulters.

The conclusion of this study is, MLP and SVM models have been trained with best hyperparameters from a optimization grid search and validated with stratified cross validation on the train dataset and then tested with the unknown test dataset to pull the performance metrics like ROC Curve, confusion matrix and classification report. Evaluated both the models with this metrices and understood that MLP model trained better in predicting the defaulters than the SVM Model.

In this study we understood the importance of the label encoding, one hot encoding, how to overcome class imbalance either by down sampling or oversampling-SMOTE. The importance of each and every parameters of the MLP and SVM model, and also understood that SVM may not perform well when the dataset has imbalanced classes or noisy features.

Way forward, since we wanted to use same dataset for model comparison I sticked to SMOTE, SVMs works better with down sampling even, can try that. And even more hyperparameter tuning can be performed on both the models.

6. Reference

[1] UCI Machine Learning Repository https://archive-beta.ics.uci.edu/dataset/350/default+of+credit+card+clients

[2] E.A. Zanaty, Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. Mathematics Dept., Computer Science Section, Faculty of Science, Sohag University, Sohag, Egypt.

[3] 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.

[4] Caruana, R., Lawrence, S. and Giles, L., 2000, November. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In NIPS (pp. 402-408).

[5] Orr, G.B. and Müller, K.R. eds., 2003. Neural networks: tricks of the trade. Springer.

[6] Obuchowski, N.A., 2003. Receiver Operating Characteristic Curves and Their Use in Radiology 1. Radiology, 229(1), pp.3-8.

Appendix 1 – glossary

Glossary:
TP – True Positive
FP – False Positive
FN – False Negative
TN – True Negative

Appendix 2 – Implementation details

Models:
In MLP and SVM the models created for this analysis are described as under,

Multilayer Perceptron Models:
Model1 - MLP baseline model using PyTorch, this model is with SMOTE and OHE
Model2 – MLP baseline model using sklearn MLPClassifier, created a baseline model with all default parameters.
Model3 – MLP hyperparameter optimization, this is the model where all the combination of the parameters are given in the grid search to find the best parameters of those combinations, where a stratified cross validation is used in the grid search. Since the combination are very large, divided the optimization into architecture and learning parameters + regularization
Model4 – MLP final model, the best parameters from the MLP hyperparameter optimized model (i.e. Model3) is incorporated in this model and provides the results out for comparison.

Support Vector machine Models:
Model5 – SVM baseline model using sklearn SVC, created a baseline model with all default parameters.
Model6 – SVM hyperparameter optimization, this is the model where all the combination of the parameters are given in the grid search to find the best parameters of those combinations, where a stratified cross validation is used in the grid search.
Model7 – SVM final model, the best parameters from the SVM hyperparameter optimized model (i.e., Model3) is incorporated in this model and provides the results out for comparison.

This is how the best parameter models – Model 4 and Model7 were arrived.