A comparative study of two machine-learning algorithms for identification of high-earners using US census data

Juan Cova and Veronica Tuffrey
City, University of London

Abstract — We report findings from a rapid comparative review of the application of two machine learning algorithms to a binary classification task using survey data. The algorithms were Multilayer Perceptron (MLP) and Support Vector Machine (SVM). Hyperparameter optimisation via grid search and 4-fold cross validations were used to identify the best models for each algorithm (those with maximum validation accuracy). Compared to the best MLP model, the best SVM model had higher accuracy (79.0% for SVM v. 78.5% for MLP), as well as a quicker processing time for model building (7.5 hours for SVM v. 19.3 hours for MLP). Our findings indicate that simpler algorithms are preferable for relatively small and uncomplicated datasets.

1 INTRODUCTION

1.1 Overview

Machine Learning (ML) has proved useful to improve data-driven decision-making in many domains. ML is "*about making computers modify or adapt their actions … so that these actions get more accurate, where accuracy is measured by how well the chosen actions reflect the correct ones*" [1]. Many of the learning algorithms that have spurred new interest in the field, such as neural networks, are based on decades old research, but ML has seen great recent growth tied to increased data availability, decreased costs of data storage, improvements in computing power, and innovation in algorithms [2]. Good decisions on effective policies and services in the health and social sector, require timely, accurate, and relevant information, where accuracy refers both to the quality of the source data, and to predictions made from these data.

With supervised learning, where an input is mapped to an output based on labelled input-output pairs, predictions can be made from source data. So we chose this approach rather than one of the other ML learning approaches (unsupervised, semi-supervised or reinforcement). This paper reports findings from our comparison of two algorithms used for supervised learning in which the classification task has binary output data. The algorithms were Multilayer Perceptron (MLP, a class of feedforward artificial neural network, and one of the most commonly used) and Support Vector Machine (SVM, a non-probabilistic classifier). We used data from the UCI (University of California, Irvine) Machine Learning Repository [3] as data from this source have been cleaned, and findings from testing algorithms can be compared to others'.

1.2 Dataset

From the UCI Repository [3] we chose the "Adult dataset" which has frequently been used to test algorithms, with findings available in published papers [e.g., 4, 5], unpublished reports [e.g., 6, 7], and in blogs [e.g., 8, 9]. Barry Becker compiled the dataset from the US 1994 Census database.

Data pre-processing before release of data into the public domain*:* Becker extracted a set of "reasonably clean records" for working adults >16 years of age. After he had removed 3,620 instances with unknown values, 45,222 instances remained. Becker converted the continuous gross income variable into a binary categorical variable, using the threshold $50,000, and the proportions were 75.22% for label 0 '<=50K' and 24.78% for label 1 '>50K' (without unknowns) [3].

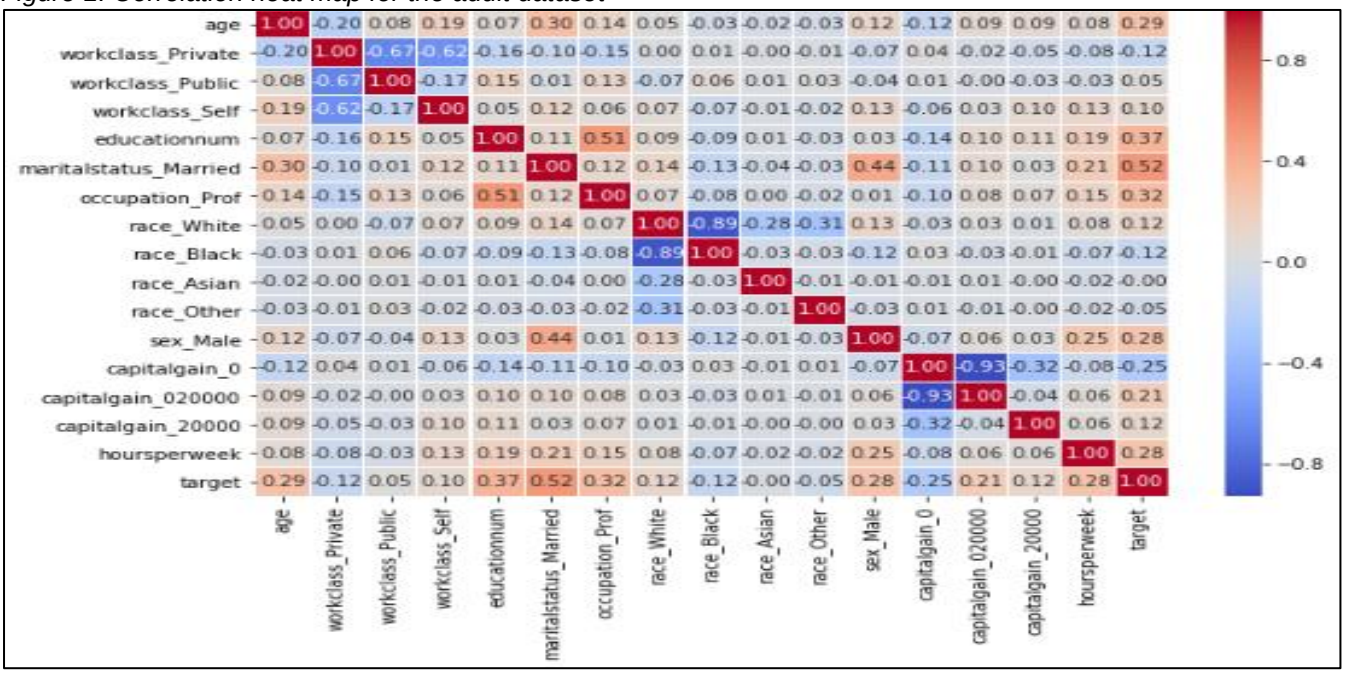
Data pre-processing by the authors*:* The original dataset had nine categorical (including the binary target variable) and six continuous variables. We removed four variables: that for weighting; the categorical education variable (because this duplicated information in the continuous education variable); the capital-loss variable (because it was highly correlated with capital gain); the relationship variable (because this was simply contextual information about the informant), and native-country (we selected the 90% of individuals whose native country was the US). We combined some of the categories for the variables workclass, marital-status, occupation and ethnicity; converted the capital gain variable into a categorial variable, and converted all string categorical variables into dummy variables with coding 1/0 because the software needed numeric variables. We standardised continuous variables so that no feature overly dominated the gradient of the loss function in MLP [10]. Finally we chose a random subset (N = 9000, with 4550 for target 0 '<=50K'; 4450 for target 1 '>50K' to balance the dataset, since unbalanced models can be biased towards the majority class and have low performance on the minority class [10]. Table 1 shows descriptive statistics

disaggregated by the two output categories. All differences between the two target categories were highly significant ($p < 0.001$) using Anova and two-sided Chi square test apart from the variable "race_Asian". The correlation heat-map (Figure 1) indicates that marital status, number of years of education, and professional occupation are the strongest predictors of high income.

Table 1: Descriptive statistics for attributes in sample dataset, disaggregated by gross income category

	Income category '<=$50K'	**Income category >$50K'**	**All sample**
Variable	**Mean (SD) and skewness**	**Mean (SD) and skewness**	**Mean (SD) and skewness**
Age	36.6 (13.4) 0.745	43.9 (10.2) 0.399	40.2 (12.5) 0.360
Years of education	9.8 (2.2) -0.229	11.6 (2.3) -0.205	10.7 (2.5) -0.121
Hours worked	39.2 (11.7) 0.258	45.8 (10.9) 0.772	42.5 (11.7) 0.375
	N in category (% of 4550)	**N in category (% of 4450)**	**% of 9000**
workclass_Private	3459 (76.0%)	2899 (65.1%)	70.6%
workclass_Public	626 (13.8%)	789 (17.7%)	15.7%
workclass_Self	465 (10.2%)	762 (17.1%)	13.6%
maritalstatus_Married	1562 (34.3%)	3808 (85.6%)	59.7%
occupation_Prof	903 (19.8%)	2219 (49.9%)	34.7%
race_White	3947 (86.7%)	4184 (94.0%)	90.3%
race_Black	494 (10.9%)	208 (4.7%)	7.8%
race_Asian	40 (0.9%)	37 (0.8%)	0.9%
race_Other	69 (1.5%)	21 (0.5%)	1.0%
sex_Male	2755 (60.5%)	3817 (85.8%)	73.0%
capitalgain_0	4355 (95.7%)	3533 (79.4%)	87.6%
capitalgain_020000	195 (4.3%)	785 (17.6%)	10.9%
capitalgain_20000	0 (0.0%)	132 (3.0%)	1.5%

Figure 1: Correlation heat map for the adult dataset



1.3 Summary of the two machine learning models

The Multi-layer Perceptron (MLP) and Support Vector Machine (SVM) are two of the most popular algorithms in modern ML. Both are applicable to classification tasks, and this was the motivation behind our choice of the two techniques. Table 2 summarises the algorithms together with pros and cons of their use.

Table 2: Use of MLPs and SVMs for classification tasks [derived from references 1, 11, 12]

MLP	**SVM**
• Neurons are arranged in layers, from the set of input nodes in the input layer, through one or more hidden layers, to the output layer. • Interconnections are allowed only between two neighbouring layers. • The network is "feedforward", that is, processing signals propagate from the input to output side.	• SVMs separate the classes with a surface that maximizes the margin between them, using boundary data points to create the decision surface. • "Kernels" can be incorporated that can make the decision boundary non-linear. • The two typical kernel functions used in SVMs are the radial basis function, and polynomial.

• The neurons use the sigmoid activation function.	
Pros: • Can handle very large datasets better than SVMs, especially if the features are very sparse. • Can handle noise better than SVMs. • More flexible than SVMs in the types of data they can support.	Pros: • Often provides impressive classification performance on reasonably sized datasets • Good for datasets with many features that are on the same scale and fewer data points • Easier than to tune and interpret compared to MLPs, as less grid searching is necessary • Fast prediction times
Cons: • Difficult to explain to a non-technical audience. • Neural networks can be slow to train. • It is a lot of work to tune the model as there are many things that can be changed, including the number of hidden units, learning rate, and initialization.	Cons: • Does not work well on extremely large datasets, since computations do not scale well as number of training examples increase, and so become computationally very expensive and lengthy. • Can be very slow if decision boundary is complicated and/or the data are very noisy. • Can be difficult to choose the most appropriate kernel for a particular application

1.4 Hypothesis statement

There are three main differentiators of learning algorithms: accuracy, speed of model building and prediction processing time. For practical applications, one may prefer a learning algorithm that builds a less accurate model fast, or a less accurate model that is quicker at making predictions.

We expected both algorithms to produce good results for accuracy of classification task on this dataset, however we expect MLP to be more accurate, following the findings of the following authors (who used different combinations of variables and hyperparameters both from ours and from each others'):

- **Khanna** [8] - who reported accuracy of 86.30% for MLP and 84.08% for SVM, and
- **Zhu** [9] - who reported accuracy of 83.28% for MLP and 83.06% for SVM.

These sources do not report speed of model building and prediction processing. However, based on the literature review summarised in Table 3, we would expect SVM to be quicker for both.

2 METHODS

Training and evaluation approach, and choice of parameters: Initially we used a randomly selected subset of data (N = 1000) to save computation time when identifying the optimal models. The final dataset contained a random balanced sample of 9,000 observations as described above in Section 1.2. Grid search was used to compare performance of algorithms using different combinations of hyperparameters and 4-fold cross validation. Table 3 summarises the hyperparameters tested.

All predictors were normalized (using the "zscore" function in Matlab), to "avoid attributes in greater numeric ranges dominating those in smaller numeric ranges" [13]. Both datasets ("data_sample" and "data_final") were randomly split into 70% training; 30% testing sets. Once we identified the optimal hyperparameters for each algorithm using validation accuracy, we used these values on the training set to build the best model for each of SVM and MLP, and then assessed the models on the test dataset using accuracy derived from confusion matrices to evaluate performance. We recorded speed of model building as a secondary measure, but not prediction speed because for both algorithms it was less than one minute. We included random seeds in our code so that the findings reported in this document could be reproduced, and also ascertained that our findings were consistent using a number of different seeds. Analysis was done using MATLAB software (Release 2018b, MathWorks Inc., Natick, Massachusetts, US); Intel Core i5 7th Gen; 2.5 GHz CPU, and 8Gb RAM.

Table 3: Choice of parameters for the two algorithms

MLP	SVM
Following Sharma & Venugopalan [13], we ran grid searches in two phases. First we tested: • *Training function*. We selected three algorithms (Gradient Descent, Conjugate Gradient and Quasi-	Following Hsu, Chang and Lin [14] we undertook grid searches in two stages, course and fine. For SVM the main decision is kernel type. As stated in Table 3, the kernel maps the observations into a

Newton) and two training functions for each algorithm, resulting into six options:
1.- Resilient Backpropagation (trainRP);
2.- Variable Learning Rate Backpropagation (trainGDX);
3.- Scaled Conjugate Gradient (trainSCG);
4.- Conjugate Gradient with Powell/Beale Restarts (trainCGB);
5.- Levenberg-Marquardt (trainLM) and
6.- BFGS Quasi-Newton (train BFG).

- *Number of layers*: We used 1, 2 & 3 hidden layers.
- *Number of neurons per layer*: Number of neurons in the first layer is determined by the number of features. For the next layers we tried 5, 10, 20, 30 & 50 neurons.

The function resulting in the maximum mean validation accuracy over the 4-fold cross validation was selected.
Then we ran a second grid search as follows:

- *Learning rate*: We considered six options: 0.7; 0.3; 0.1; 0.01; 0.001; 0.0005.

To prevent overfitting, we used early stopping instead of stopping criteria.
We trained the model until validation errors started to increase, allowing a maximum of six validation failures.

higher dimension space so as to be more easily linearly separable. Standard kernel types are:

- *Linear*: With linear kernel the only hyper-parameter to be tuned is cost function C.
- *Polynomial*: With polynomial kernel, degree must be tuned as well as cost function.
- *Radial*: For radial ("Gaussian") kernel, γ ("width" or "kernel scale") must be tuned as well as C.

To identify the range of hyperparameter values to be included in a finer grid search, we first ran automatic hyperparameter optimization and considered 70 objective function evaluations which included all eligible parameters available.
The best two evaluations (those with minimum validation loss) for each of the three kernel types were used to implement the finer grid search. For this, two additional hyperparameter values were considered for each kernel type, interpolated between the optimal values identified from the automatic optimization. The model with the maximum mean validation accuracy was selected.

3 FINDINGS

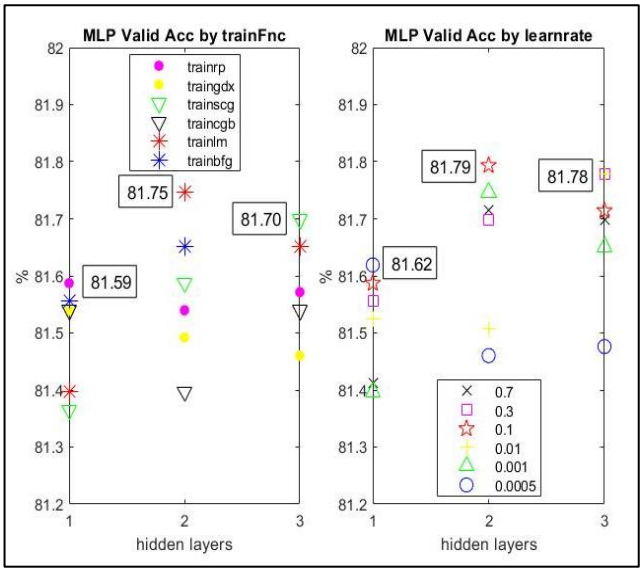
Tables 4 to 6 and Figures 2 to 5 present the experimental results, and these are discussed in Section 4.

Table 4 (below left): MLP processing time by training function and number of hidden layers.

Table 5 (below right): MLP processing time using "trainLM" training function, by learning rate and number of hidden layers[1]

		Time (minutes)		
Algorithm	Train Funct.	1 hidlay	2 hidlay	3 hidlay
Gradient Descent	trainRP	0.16	0.81	6.84
	trainGDX	0.19	1.57	14.92
Conjugate Gradient	trainSCG	0.12	0.96	10.77
	trainCGB	0.15	1.42	17.75
Quasi Newton	trainLM	0.23	7.83	152.11
	trainBFG	0.41	43.67	-
All	All	1.25	56.26	-

	Time (minutes)		(hours)
LearnRate	1 hidlay	2 hidlay	3 hidlay
0.7	0.38	9.84	2.68
0.3	0.36	10.64	2.60
0.1	0.34	10.39	2.52
0.01	0.32	11.87	2.57
0.001	0.29	11.91	2.66
0.0005	0.31	12.59	2.75
All	2.01	67.24	15.79



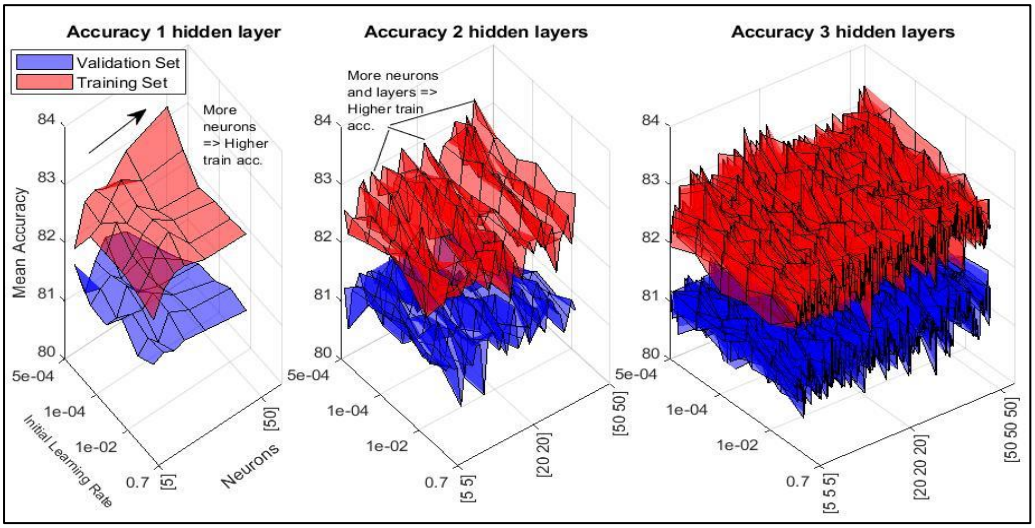


Figure 2 (above left): MLP validation accuracy by training function and number of hidden layers

Figure 3 (above right): MLP validation accuracy by learning rate and number of hidden layers

Figure 4 (left): 3D plot for MLP validation and training accuracy, showing association of accuracy with number of hidden layers, number of neurons, and initial learning rate

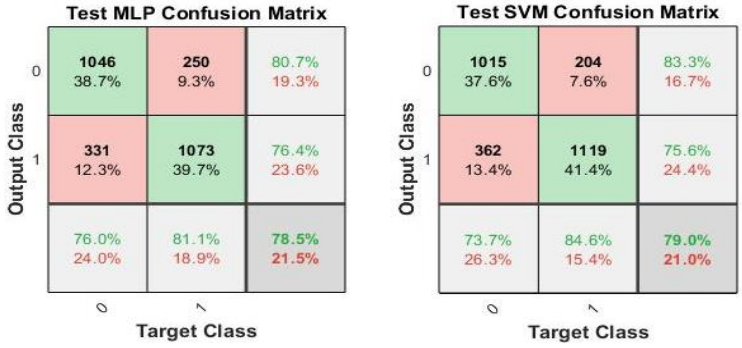
[1] The sixth function "trainBFG" was not trialled with three layers due to exponential increase in processing times.

Kernel Function	Box Cons-traint	Kernel Scale / Polynomial Degree	Validation Accuracy	Time (minutes)
Linear*	0.04	-	80.76%	0.03
Linear	1.64	-	80.88%	0.21
Linear	3.24	-	80.91%	0.47
Linear*	4.84	-	80.91%	0.71
Gaussian	162.39	6.35	81.38%	0.48
Gaussian	213.84	6.35	81.50%	0.96
Gaussian	265.29	6.35	81.42%	1.09
Gaussian*	316.74	6.35	81.42%	1.31
Gaussian	162.39	10.67	81.43%	0.75
Gaussian	213.84	10.67	81.38%	0.39
Gaussian	265.29	10.67	81.35%	0.52
Gaussian	316.74	10.67	81.46%	0.59
Gaussian	162.39	14.99	81.64%	0.37
Gaussian	213.84	14.99	81.63%	0.23
Gaussian	265.29	14.99	81.53%	0.26
Gaussian	316.74	14.99	81.47%	0.32
Gaussian*	**162.39**	**19.31**	**81.85%**	0.23
Gaussian	213.84	19.31	81.75%	0.17
Gaussian	265.29	19.31	81.72%	0.21
Gaussian	316.74	19.31	81.71%	0.23
Polynomial*	0.52	2	81.65%	2.43
Polynomial	0.53	2	81.65%	4.24
Polynomial	0.53	2	81.65%	4.33
Polynomial*	0.54	2	81.65%	4.32
Polynomial	0.52	3	81.00%	8.20
Polynomial	0.53	3	81.06%	10.17
Polynomial	0.53	3	81.18%	10.15
Polynomial	0.54	3	81.11%	10.37
Polynomial	0.52	4	77.67%	10.29
Polynomial	0.53	4	76.59%	10.09
Polynomial	0.53	4	76.86%	10.05
Polynomial	0.54	4	75.79%	10.03

* Hyperparameters identified via automatic optimisation

Table 6 (left): Findings from SVM hyperparameter grid search

Figure 5 (below): Confusion matrices for MLP and SVM from test data



4 ANALYSIS & CRITICAL EVALUATION OF RESULTS

First grid search for MLP: Figure 2 shows the validation accuracy for each training function using "data_final". Values were similar for all functions considered, whether for 1, 2 or 3 hidden layers, but accuracy was greatest with the Levenberg-Marquardt (trainLM) function with two hidden layers. Table 3 shows training times, which were fastest for the Gradient Descent and Conjugate Gradient algorithms, while the Quasi-Newton functions were slower, especially the "trainBFG" option.

We tested various combinations of neurons, hidden layers and training functions, and found as the number of neurons and layers increased, processing time was greater. Comparison of validation accuracy between different combinations of neurons and hidden layers showed that "trainLM" consistently resulted in higher values, and "trainGDX" in smaller ones. Quasi-Newton algorithms consistently had longer processing times, and "trainBFG" was by far the slowest algorithm. (Figures and Tables are not provided here due to lack of space, but findings were consistent with Figure 2 and Table 4). Testing the code on Data_Sample led to 2 – 3% higher values for errors and shorter processing times - the latter was especially the case for SVM (small sample took SVM 6 minutes, and MLP 5.9 hours[2] v. full sample took SVM 7.5 hours, and MLP 19.3 hours).

Second grid search for MLP*:* Using the best training function "trainLM", we tuned over the initial value for the learning rate, using values between 0.7 and 0.0005. Figure 3 shows validation accuracies for each rate, which were similar for all rates considered, whether with 1, 2 or 3 hidden layers. For one hidden layer the optimal learning rate was 0.0005, with two layers it was 0.1 and with 3 layers it was 0.01 and 0.3 (same values). Table 5 shows processing times in minutes using the trainLM function. Although learning rate is said to be the most important hyperparameter to tune for neural networks [15], we found only a small effect of learning rate on processing time and accuracy. Minimum processing time was observed for different values of learning rate according to number of layers (0.001 for one, 0.7 for two and 0.1 for three layers).

Tuning over "learnrate" helped increase accuracy from 81.59 to 81.62% for one layer; from 81.75 to 81.79% for two layers and from 81.70 to 81.78% for three hidden layers. We selected our optimal MLP model as that with two hidden layers of 10 and 30 neurons, Levenberg-Marquardt training function and learning rate of 0.1. With validation accuracy of 81.79% the accuracy of this model is 0.01% greater than with three hidden layers, and 0.27% greater than with one hidden layer, while the processing time is still measured in minutes not hours (Table 5). Figure 4 shows training and validation accuracy considering one, two and three hidden layers, over the combination of neurons and initial learning rates tested. The figure shows that adding more neurons and/or layers tends to increase training accuracy but not validation accuracy. In fact, best validation accuracy was found using [20], [10 30] and [5 5 20] neurons while best training accuracy was found using [50 30] and [30 50 50] neurons.

Grid search for SVM: Table 6 shows the findings from the fine grid search for SVM, using the optimal values obtained from the automatic optimisation (with astericks) and values interpolated between them. Findings from polynomial degrees 3 and 4 are included in the table to show how as polynomial degree of the SVM kernel increases, accuracy decreases while processing time increases, for this dataset. The best

[2] SVM: 4.7 mins automatic optimization + 1.3 mins finer grid search; MLP: 1.2 hrs 1st grid search + 4.7 hours 2nd grid search.

model used radial function for the kernel, with width 14.99 and Box constraint C =162.39. C is a regularization parameter that helps prevent overfitting – the value here is quite large indicating low regularization, and heavy weighting given to misclassification. High values increase training times, however this was not a concern for us because our dataset was relatively small.

Comparison of the optimal models for MLP and SVM: The confusion matrix in Figure 5 compares the values for accuracy of MLP and SVM. This shows that, contrary to our initial hypothesis, the best SVM model had slightly higher accuracy (79.0% for SVM v. 78.5% for MLP). SVM also had lower computation time (7.5 for SVM v. 19.3 hours for MLP)[3]. For this study, we used accuracy as our primary comparator, but if the practical application of the algorithm meant it was most important to identify high-earners (for example for tax collection purposes) then the priority would be minimising false negatives, and one would use a measure like sensitivity. Also, values of classification accuracy are not reliable if the dataset is imbalanced.

The accuracies of the two algorithms were almost identical, but applying the concept of "Occam's razor" (whereby if there is more than one suitable algorithm with comparable trade-offs, the one that is least complex to deploy and easiest to interpret should be chosen), SVM is our preferred algorithm for this task.

5 CONCLUSIONS

Contrary to our initial hypothesis, we found with this dataset, SVM was a preferable algorithm with respect to accuracy, as well as speed of model building. These findings cannot be generalised to datasets of different size, complexity, ratio of continuous to categorical variables, and ratio of attributes to samples. Additional factors must be considered when choosing an algorithm, including transparency and "interpretability" [2]. In this respect, SVM is also preferable to MLP.

A lesson learnt from the model selection process is that the automatic optimisation function in Matlab for SVM is robust. Our fine grid search did not improve the optimal result obtained from the automatic option. Useful future work on this dataset would be to select the most important predictors and compare the algorithms with a more restricted dataset, to examine trade-off between training time and accuracy.

Our study provides a valuable template for comparing the performance of other algorithms.

6 REFERENCES

[1] Marsland S. *Machine learning: an algorithmic perspective.* Florida, US: Chapman and Hall/CRC; 2011.

[2] Internet Society. *Artificial Intelligence and Machine Learning***.** Geneva: The Internet Society; 2017.

[3] Dua D, Graff C. 2019. *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml].

[4] Chakrabarty N, Biswas S. A statistical approach to adult census income level prediction. *arXiv preprint arXiv:181010076* 2018.

[5] Ali M, Ali HM. Performance Analysis of Classification Learning Methods on Large Dataset using two Data Mining Tools. *Journal of Independent Studies and Research – Computing* 2015; 13 8 - 15.

[6] Chockalingam V, Shah S, Shaw R. Income Classification using Adult Census Data. *Unpublished report* 2017.

[7] Hongzheng H. Classifying income potential from the adult dataset. *Unpublished report* 2018.

[8] Khanna R. 2018. *Comparative Study of Classifiers in predicting the Income Range of a person from a census data* [https://towardsdatascience.com/@ritvikkhanna09?source=user_popover].

[9] Zhu H. 2016. *Predicting Earning Potential using Adult Dataset* [https://rpubs.com/H_Zhu/235617].

[10] Burkov A. *The hundred-page machine learning book.* 2018.

[11] Salankar SS, Patre BM. SVM Based Model as an Optimal Classifier for the Classification of Sonar Signals *WASET International Journal of Electronics and Communication Engineering* 2007; 1.

[12] Zanaty EA. Support vector machines (SVMs) versus multilayer perception (MLP) in data classification. *Egyptian Informatics Journal* 2012; 13**:**177-183.

[13] Sharma B, Venugopalan K. Comparison of neural network training functions for hematoma classification in brain CT images. *IOSR Journal of Computer Engineering* 2014; 16**:**31-35.

[14] Hsu C-W, Chang C-C, Lin C-J. 2016. *A practical guide to support vector classification* [https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf].

[15] Bengio Y. *Practical Recommendations for Gradient-based Training of Deep Architectures***.** In Neural Networks: Tricks of the Trade, 2nd ed Eds: Montavon, G et al. Heidelberg: Springer; 2012: 437-478

[3] SVM: 5.8 hrs automatic optimization + 1.7 hrs finer grid search v. MLP: 3.4 hrs 1st grid search + 15.8 hrs 2nd grid search.