A Comparison of Neural Networks and Support Vector Machines Algorithm Performance in Human Resource Analytics

Hannes Draxl, Ryan Nazareth

1. Introduction, motivation and hypothesis statement

Employee turnover is not only a cumbersome process but also a financial burden. Employers can benefit from knowledge of an employee's likelihood of changing the company. Often, decisions for leaving a company do not emerge out of a sudden but are usually the outcome of careful pre-planning. Utilising a data-driven approach to analyse which employees are likely to leave the company soon, therefore, can be of great interest to managers. With the gained knowledge, human resource management teams can act upon these predictions to persuade employees to stay before they jump ship. The main aim of this study is to critically compare and contrast the performance of a feedforward artificial neural network, specifically a multi-layer perceptron (MLP) and a support vector machine (SVM) classifier, in predicting if an employee will leave the company or not.
Our Hypothesis is that whilst we expect both MLP and SVM to perform well on the chosen data set, we hypothesise that MLP will outperform SVM based on the fact that MLP might be able to model more complex patterns compared to the kernel functions of SVM.

2. Dataset description, analysis and preparation

The dataset is part of a Kaggle challenge [7] and consists of 10 variables being a mixture of numerical and categorical variables. Table 1 depicts the individual features, their mean and standard deviation as well as categorical features and a target variable "Left" which consists of the two classes *stay* and *leave*. Furthermore, this dataset has ≈ 15.000 samples with a class imbalance of 11428 (stayed) and 3571 samples corresponding to "left".

Table 1 Variable description

Variable	Scaling	Mean	Standard Dev
Satisfaction level	Numerical	0.61	0.25
Last evaluation	Numerical	0.76	0.17
Number of Projects	Numerical	3.8	1.23
Avg. Monthly Hours	Numerical	201	50
Time Spend Company	Numerical	3.5	1.46
Work Accident	Numerical	0.14	0.35
Promotion last 5 years	Numerical	0.02	0.144
Sales (10 levels)	Categorical	N/A	N/A
Salary (low, med, high)	Categorical	N/A	N/A
Left (target variable)	Categorical	N/A	N/A

As a next step, the dataset was explored by visualising a correlation heat map to gain a first intuition of predictive capability of our features. The correlation heat map in figure 1a displays the correlation between features themselves and the target variable. Most of the features show only very little correlation with each other as well as with our "left" target variable. One strong correlation can be observed between the target variable and the satisfaction level of approximately -0.5. Figure 1b depicts violin plots of the satisfaction level grouped by the *salary* and *promotion_last_5years* features. The yellow colour represents the employees which left the company. From this plot, we can conclude that the salary level is not a strong predictor but that some features such as *salary* and *promotion_last_5years* together can enhance our model's predictive capability.

From Table 1 it can be observed that some of the features, such as the Avg. hours per month lies on a much higher scale than the rest of the features. Therefore, the features were standardised to $\mu = 0$ and $\sigma = 1$ to make sure that no feature is skewing the algorithm results towards the features on higher scales. Moreover, the two categorical features "salary" and "sales" had to be one-hot encoded and concatenated onto the numerical feature matrix. As a result, the feature matrix consists of shape *14999*x*21*.

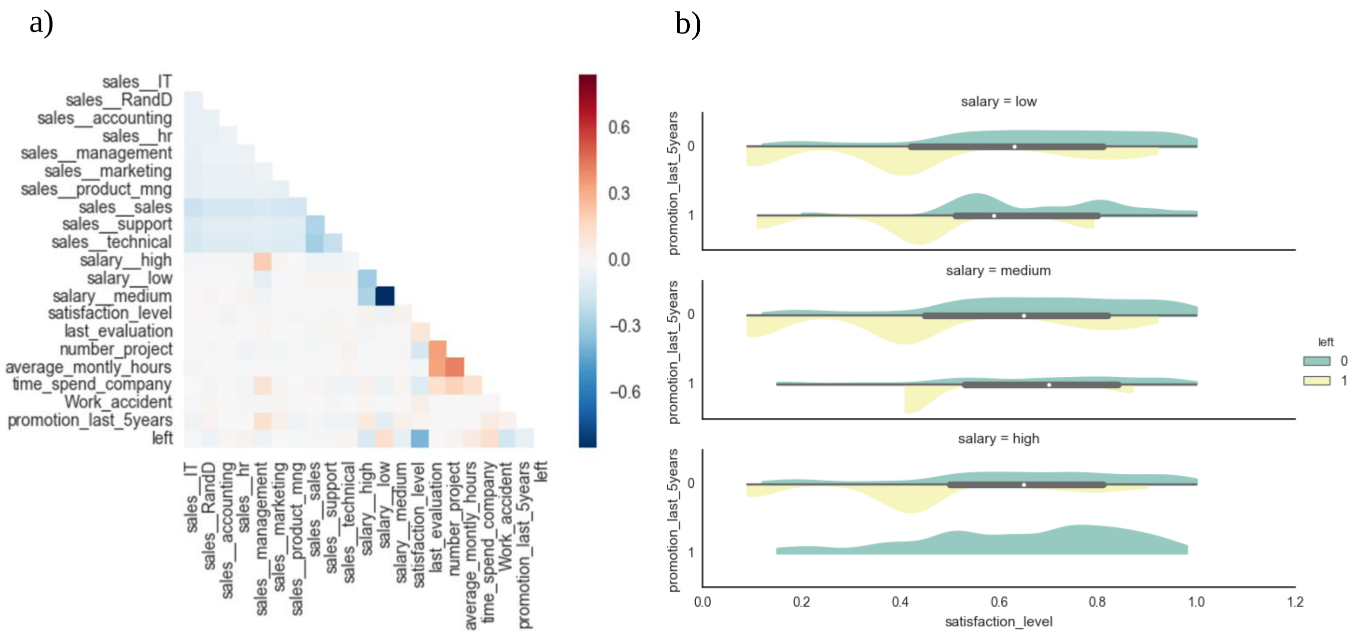


Figure 1: a): Correlation Heat map between features and the target variable (last column/last row). b): A violin plot displaying the distribution of the satisfaction level grouped by salary levels and possible promotions over the last 5 years. The green colour represents that the employee stayed with the company.

3. Neural Net Methods

The following section describes the two neural network models and the pros and cons of each, followed by an overview of the hyperparameter (HP) tuning and evaluation pipeline used in the study.
The choice of a given model depends on the dataset in question. There are several advantages and disadvantages of using each of the models, which have been summarised in the table below [12][6][4]:

	SVM	**MLP**
Pros	Effective for high dimensional problems	Unsupervised feature learning through hidden layers and non-linear activation functions
	Memory efficient: uses a few training points (support vectors) for computing the decision boundary	Powerful algorithm for numerous different kinds of data sets, especially perceptual data
Cons	Does not perform well if there is too much noise in the dataset	Requires large amounts of data and computational resources
	Depending on the chosen kernel, training and testing time can be very slow	Computationally expensive as a number of hyperparameters need to be tuned

Table 2: Summary of the pros and cons of SVM and MLP algorithms [12][6][4]

SVM is a supervised algorithm used for classification and regression tasks. A decision boundary is computed which maximises the distance from the data points closest to the decision boundary, also known as support vectors [12]. Given a training set, the optimal decision boundary is computed which separates the classes with a geometric margin or ‘gap’ such that it results in good and confident predictions on the samples [12]. MLP is a supervised feedforward learning algorithm that learns functions in an unsupervised fashion through its hidden layers and activation functions to map the input in the training data to an output [6] [13]. The hidden layer transforms the values from the inputs neurons in the leftmost layer with a weighted linear summation followed by an activation function which maps the values [9]. These values are received by the output layer which transforms them into output values by utilising a linear combination (regression) or a sigmoid function (binary classification) [6]. Through backpropagation, we gain the partial derivatives of our loss function w.r.t. the weights and use these to adjusts the weights through gradient descent. This process is repeated over a specified number of epochs and until we reach a certain minimum of the loss function [6].

4. Training and evaluation methodology

Modelling was carried out in MATLAB (version 2016b) using the functions in the Neural Network Toolbox [9] for MLP implementation and Statistics and Machine Learning Toolbox [10] for SVM implementation. As a first step, we split our data into 70% training and 30% test set which is important to evaluate the generalisation performance in an unbiased way. After this, for both SVM and MLP, we applied grid search for hyper parameter tuning embedded in a 5-fold stratified cross validation. Throughout the grid search procedure, we used misclassification rate as a performance metric provided by the grid search hyper parameter tuning function. After grid search, the models with the optimal HP were retrained on the full training data. For the test set evaluation, we relied on a confusion matrix, and especially the recall score, which is helpful in our case of identifying how many samples our models can correctly classify as left out of all samples that correspond to "left" (the true positive rate).

4.1 Grid Search

Following the data pre-processing steps discussed in the previous section, both SVM and MLP were tuned with their distinct HP values. For SVM, the tuning process was split into distinct steps to reduce the amount of computing needed for HP tuning: First, grid search was applied over different kernel functions (linear, Gaussian, polynomial). Subsequently, randomised grid search was utilised to tune the specific kernel parameters C (Box Constraint) and σ (Kernel Scale) over a range of possible parameter values displayed in Table 3. We chose to utilise randomised search [1] to speed up the the enormous compute time needed to tune these parameters. Also, from initial experiments, we could observe that close values around the default parameters of C and σ worked best. For both C and σ, we chose therefore to tune between a range of 0.1 and 1 with a step size of 0.1 (see Table 3 for a summary of the tuning parameters).

Such as with SVM, a HP grid search was conducted with MLP as well. The Neural Computing Toolbox [9] does not implement a grid search so this had to be coded in manually. The first two parameters set the number of hidden neurons in each layer and the number of layers. The higher this value, the more capacity the network has to model complex input distributions [6]. A range of values have been chosen for the learning rate. The smaller the learning rate, the longer it takes to reach the minimum of the error function [6]. Conversely, if the learning rate is too large, there is a possibility of the weight updates diverging from the error minimum. During the HP tuning of the MLP, a fast algorithm based on Scaled Conjugate Gradient (SCG) backpropagation method [11], developed by Moller, for updating weights and bias values was chosen. The traditional backpropagation algorithms, adjust the weights in the direction the gradient is decreasing the fastest. This does not necessarily result in fastest convergence. By searching along the conjugate direction compared to the steepest gradient direction, quicker convergence is achieved [11]. The error in classification is minimised using the mean square error. Furthermore, early stopping is also implemented to stop training before the maximum number of epochs is reached [2]. This prevents the algorithm from overfitting on the training data and improves generalisation [2]. In this case, the algorithm stops if the validation error does not decrease for 6 consecutive epochs. See Table 3 for the chosen HP tuning range of the MLP.

SVM	MLP
• Kernel function (first round): [linear, Gaussian, polynomial] • Box Constraint (second round): [0.1 – 1] • Kernel Scale (second round): [0.1 – 1]	• Number of hidden neurons: [60, 100] • Number of layers: [3, 7] • Learning rate: [0.01, 0.1, 0.9]

Table 3: Tuned hyperparameters of SVM & MLP

5. Choice of parameters and experimental results

5.1 Hyper parameter optimisation results

Both SVM and MLP were tuned on the defined grids described in the previous section. For SVM, the best performing kernel was the Gaussian kernel (with default values for C and σ of 1) and resulted in a very high training accuracy score of 98.9%. The linear kernel (accuracy of 78%) was far behind the Gaussian and polynomial kernel (95% accuracy). Therefore, we continued to tune the SVM with Gaussian kernel further in its box constraint (C) parameter and kernel size (σ) with a random search using 5-fold cross validation and 10 random picks. From Table 4 we can observe, that randomised grid search resulted in a σ of 0.16 and a C parameter value of 0.28. This setting, however, led to a strong decrease in accuracy score of 86%. As such, the default values of 1, for σ and C (which resulted in a training accuracy score of 97%) were chosen as the parameters for retraining SVM on the full training set leading to an accuracy score of 98.8%.

For the MLP, the optimal parameters were found to be: Number of neurons 60, number of layers: 3 and learning rate: 0.1. MLP was retrained with a more robust backpropagation method invented by MacKay in 1992 [8] which is computationally more expensive but more accurate than the SCG method [5]. This network function performs Bayesian Regularisation (BR) [5] [8] by implementing the Levenberg-Marquardt algorithm [5], to minimise the weights and biases so that the model generalises well. An early stopping criterion was set so that the training terminates if the validation error continues to increase for a certain number of iterations [2]. The validation and training errors normally decrease during initial training but as the network starts to overfit, the validation error increases. In such a case, training would stop once a certain number of iterations have passed [2]. In figure 2, the best validation performance (mean square error: 0.015), was recorded at epoch 493 when the training stopped before the maximum number of epochs (500) were reached.

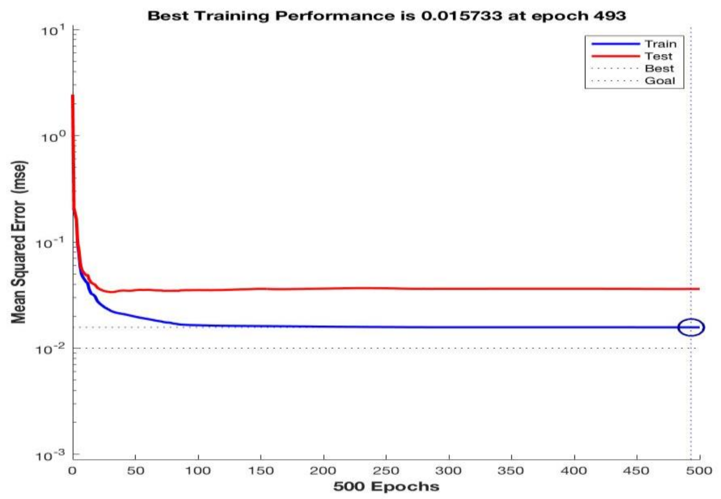


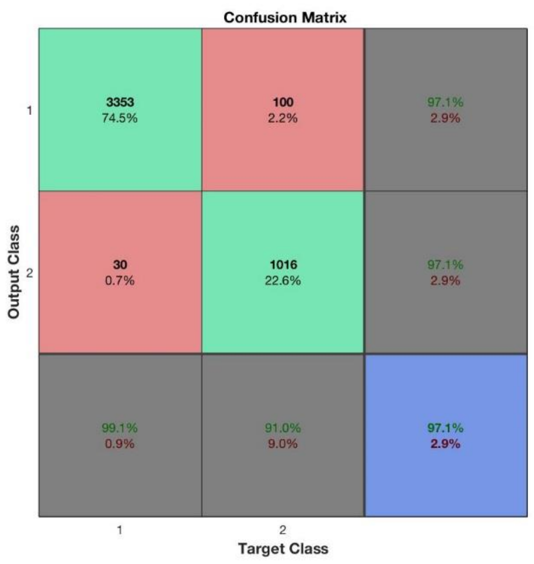
Figure 2: Performance curve for the optimally tuned MLP model on the training data. The algorithm stops when the maximum number of epochs are reached if none of the other stopping criteria are satisfied before this.

Figure 2 shows the results of the training and validation error curve when re-training the optimally tuned MLP on the training dataset. The mean square error from the training set decreases with the number of epochs whilst the error on the validation set decreases up to epoch 50, and then increases relative to the training set. In this case, the early stopping criterion was not satisfied since the validation error remained relatively consistent through this training run. Overall, we achieved a training accuracy of 98.5%. Although the cross-entropy "performance" metric is preferred for this type of problem, both the mean square error and cross entropy performance metrics gave identical values (0.0184).

5.2) Test set evaluation

In our case, the recall rate is of special interest because we want to mitigate the effect of misclassifying people as “not left”. As such, we set the focus on comparing the two test set results based on their recall rates. An illustration of the confusion matrices for both models is shown in figure 3. SVM resulted in a recall rate of 91% whilst MLP led to 93.3%, an important improvement. For precision, we can see the reverse effect with a precision rate of 97.1% for SVM vs 93.8% for MLP. The final test set accuracy scores can also be observed in the confusion matrices (in the blue square) and shows that in terms of accuracy, SVM is better with 97.1% compared to MLP of 96.8%.

a)



b)

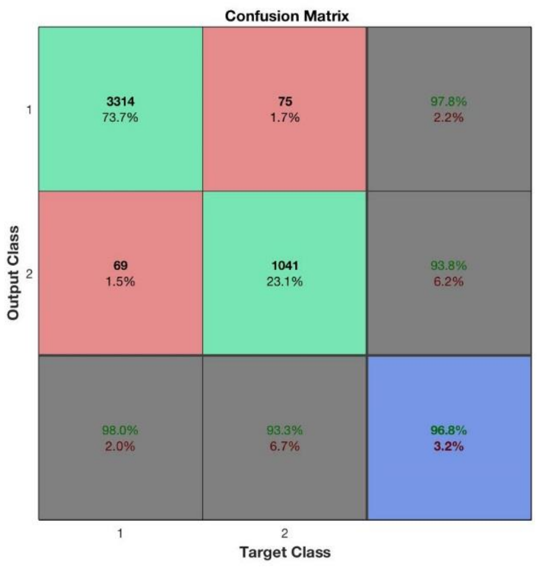


Figure 3: a): Confusion Matrix SVM. b): Confusion Matrix MLP

6. Analysis and critical evaluation of results

In essence, both algorithms accomplished a very high and comparable performance on this data set to classify employee departure. For the choice of HP in this experiment, the training time was a big issue for both SVM and MLP. Splitting the hyperparameter search of SVM into two phases was an important decision to reduce the compute time. An alternative way of evaluating if a Gaussian kernel might be more suitable than a linear kernel could be done through investigating a 2D PCA plot. If the decision boundary is highly non-linear, we could have ruled out the linear kernel right from the beginning and saved some time tuning over different kernel functions. The second tuning round, on a random grid with only 10 individual combinations of values on 5-fold cross validation, took around 2 hours. In the end, random search did not find any better values for σ and C but this is only due to the fact, the restricted the random search procedure to only 10 random combinations. A larger number of combinations would probably result in even better HP compared to the default values, which were better than the random values found by the tuning process. However, considering that computational power was limited, this was not possible for us to explore. We also evaluated our results in Python using Sci-kit learn [13] and found that tuning was much faster compared to the MATLAB implementation. HP tuning has an even greater effect on MLP compared to SVM due to the wider range of parameters that can be tuned. In addition, the choice of backpropagation algorithm during the tuning process can make a huge difference to training time. The SCG method implemented here takes approximately 30 seconds for tuning one set of HP (including 5-fold cross validation). For the 12 combinations in this experiment, this took approximately 6 minutes. In contrast, BR took around 10 minutes for training one run of the optimally tuned HP on the training dataset. Although BR gives more accurate results, it would have taken around 6 hours if used for tuning instead of SCG, which is not practical.

In terms of the final results, we are not surprised that MLP had a higher recall rate compared to SVM. Reducing false negatives has a great importance for this kind of problem set as we do not want to overlook key employees leaving the company. We suppose that the capabilities of learning higher level features through its hidden layers and non-linear activations are what gave it an edge compared to SVM. A limitation of our study was that we were not able to test the effect of momentum as one of our tuning parameters since it is not implemented in the BR algorithm chosen for this study. Momentum usually allows faster convergence and prevents a network from getting stuck in a local minimum [6] [9].

7. Conclusions and Future Work

To conclude, both SVM, as well as MLP, led to strong performance in predicting if a customer will leave the company or not and as such, this has a high value for human resource departments. Throughout the analysis, SVM and MLP showed their strengths and weaknesses, being powerful algorithms but requiring great amounts of computing and time to tune and evaluate. In terms of results, SVM outperformed MLP in terms of the overall test accuracy (F1 score) but MLP had a slightly higher recall rate compared to SVM which is in line with our initial hypothesis.

Further work could include extending the MLP to convolutional neural networks and using Self-organising maps to reduce dimensionality and look at cluster patterns [6] [9]. Also, given the slight class imbalance of our training set, we could utilise Synthetic Minority Over-Sampling Technique [3]. This would oversample the rarer classes (by generating synthetic samples) to produce a more even distribution of the data and might lead to a further performance boost.

8. References

[1] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.
[2] Caruana, R., Lawrence, S. and Giles, L., 2000, November. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In NIPS (pp. 402-408).
[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] Description of MLP algorithm and diagram in Scikit-learn
http://scikit- learn.org/stable/modules/neural_networks_supervised.html
[5] Foresee, F.D. and Hagan, M.T., 1997, June. Gauss-Newton approximation to Bayesian learning. In Neural Networks, 1997., International Conference on (Vol. 3, pp. 1930-1935). IEEE.
[6] Haykin, S. S., 1999, Neural networks: a comprehensive foundation, 2nd edn, Prentice Hall, Upper Saddle River, N.J.
[7] Kaggle dataset, https://www.kaggle.com/ludobenistant/hr-analytics
[8] MacKay, D.J., 1992. Bayesian interpolation. Neural computation, 4(3), pp.415-447.
[9] MATLAB Neural Network Toolbox Documentation, https://uk.mathworks.com/help/nnet/
[10] MATLAB Statistics and Machine Learning Toolbox, https://uk.mathworks.com/products/statistics.html
[11] Møller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural networks, 6(4), pp.525-533.
[12] Ng, Andrew, 2015, CS229 Machine Learning lecture note 3, from: http://cs229.stanford.edu/materials.html accessed March 2017
[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), pp.2825-2830.