Comparison of Multilayer Perceptron and Support Vector Machine Models for Binary Classification on Wine Quality Dataset

Shawn Ban, Shalini Leelananda

1. Introduction, description and motivation

This project critically evaluates two models for predicting if a wine is pleasurable (rating 7 and over) or not, based on the wine's physicochemical properties. We compare a feedforward Multilayer Perceptron (MLP) and a Support Vector Machine (SVM). The motivation for using these models is that both models can perform a **binary classification** task, but achieve it through vastly different approaches. We investigate various parameters of these models and different evaluation criteria to assess the accurate classification of wine.

Models that use physicochemical properties to predict wine quality could be valuable for wine merchants looking to buy wine, or growers who seek to improve the quality of their crop.

2. Dataset

The dataset used in this project relates to white variants of Portuguese "Vinho Verde" wine, was previously created by Cortez, et al. [1] and was obtained from the UCI Machine Learning Repository [2]. The dataset contains 4898 observations with 11 predictor variables and one response variable. The predictor variables are continuous and represent physicochemical properties of the wine samples, while the response variable represents wine quality on a scale of 0 to 10. For this project, we classify wines with a rating of 6 and below as poor wines (class 1) and those with a rating of 7 and above as good wines (class 2) leaving us with a binary classification problem. There are 1060 observations of good wine, about 21.6% of the total, and 3838 observations of poor wine, about 78.4% of the total. As the imbalance between classes is not too severe, we have chosen not to use resampling techniques such as SMOTE [3].

We performed a **Z-score normalization** on the input variables for better comparison between variables. This way, the models could evaluate all inputs with similar significance and prevent any single variable from dominating the model weights. As can be seen from Figure 1 after normalization, good wine has a lower mean than poor wine for the first eight variables, and a higher mean for the last three variables.

	Before Normalization				After Normalization			
	Good Wine (1060 obs)		Poor Wine (3838 obs)		Good Wine (1060 obs)		Poor Wine (3838 obs)	
Features	**Mean**	**St Dev**	**Mean**	**St Dev**	**Mean**	**St Dev**	**Mean**	**St Dev**
fixed acidity	6.725	0.769	6.891	0.860	-0.154	0.911	0.042	1.019
volatile acidity	0.265	0.094	0.282	0.102	-0.128	0.934	0.035	1.015
citric acid	0.326	0.080	0.336	0.130	-0.067	0.663	0.019	1.074
residual sugar	5.262	4.291	6.703	5.225	-0.223	0.846	0.062	1.030
chlorides	0.038	0.011	0.048	0.024	-0.348	0.510	0.096	1.078
free sulfur dioxide	34.550	13.797	35.517	17.788	-0.045	0.811	0.012	1.046
total sulfur dioxide	125.245	32.725	141.983	44.145	-0.309	0.770	0.085	1.039
density	0.992	0.003	0.994	0.003	-0.540	0.927	0.149	0.968
pH	3.215	0.157	3.181	0.148	0.178	1.041	-0.049	0.983
sulphates	0.500	0.133	0.487	0.108	0.090	1.166	-0.025	0.948
alcohol	11.416	1.255	10.265	1.101	0.733	1.020	-0.202	0.894

Fig. 1. Summary statistics before and after normalization.

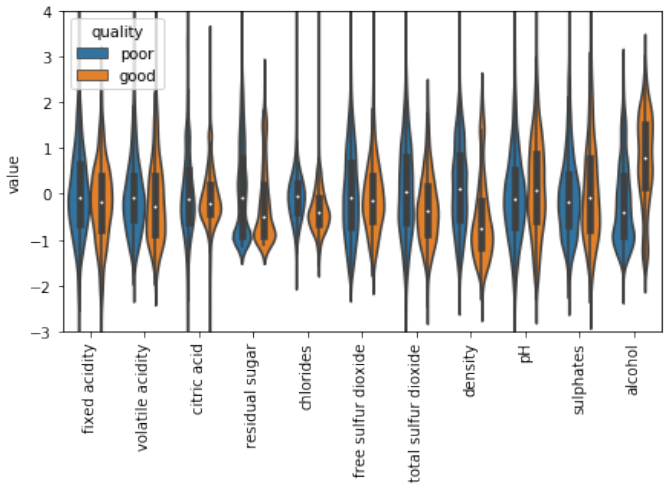

Fig. 2. Distribution of input variables by class.

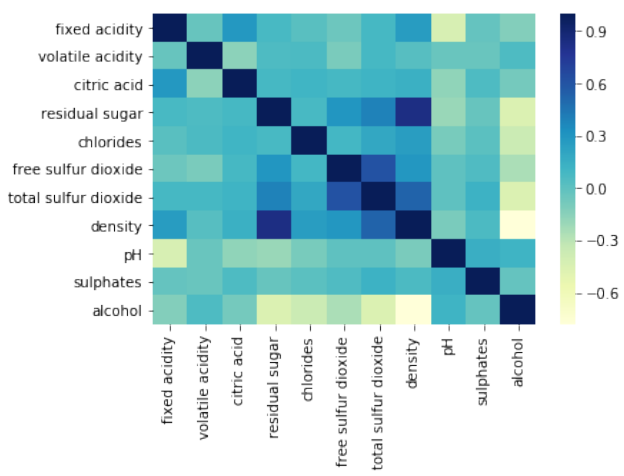

Fig. 3. Correlation heat map of input variables.

The distributions of the input variables of the two classes of wine are shown in Figure 2. We also examined the correlation between the various input variables (Figure 3). As most of the correlations were below 0.50 and the dimensionality of the dataset was low, we chose to retain all the variables as inputs for our models. At this preliminary stage, it is difficult to determine which features carry more importance, although the difference in density and alcohol between the two classes is noteworthy.

3. Model summaries with pros and cons

3.1 Multilayer Perceptron (MLP)

The MLP is one of the most commonly used neural network models, with neurons arranged in layers from input layer, to hidden layers and an output layers. Neurons are activated by a sigmoid function, and the network is feed-forward, with signals propagating from input to the output [4]. Training uses the backpropagation algorithm, where the magnitude of the error signal determines to what degree the weights in the network should be adjusted such that the combination of weights with the smallest error is found [5]. The algorithm also uses this procedure to calculate the local gradient of the error surface and changes the weights in the direction of steepest local gradient in an attempt to converge to the absolute (or global) minimum of the error surface [6].

An **advantage** of MLPs is it does not require any prior assumptions regarding the distribution of training data. Another benefit of the multilayer perceptron approach is that no decision regarding the relative importance of the various input measurements need to be made; during training the weights are adjusted to select the most discriminating input measurements [6].

A **disadvantage** of using an MLP is deciding on the network architecture, namely the number of layers and nodes in those layers. A network with too many nodes and hidden layers is highly apparent to eventually learn all the training patterns in the training data leading to overfitting. In a problem with noisy data the network would eventually learn all the noise in the training data [6]. Decisions also need to be made in choosing suitable values for the learning rate and momentum parameters required by the backpropagation training algorithm. More advanced algorithms avoid the need to specify such sensitive training parameters [7].

3.2 Support Vector Machines (SVM)

Support Vector Machines classify data by choosing a hyperplane that provides the largest separation, or margin, between different classes. This hyperplane represents a decision boundary, and the distance of each data point to the decision boundary reflects the confidence of the prediction on that data point. Intuitively, for a greater margin or separation, we expect a lower generalization error. The SVM algorithm uses a subset of the training data, the support vectors, to find the **maximum-margin hyperplane**. This yields a quadratic optimization problem, one of minimizing a convex quadratic function subject to linear constraints. Cortes and Vapnik further applied kernel functions to create nonlinear classifiers [8], allowing the SVM classifier to fit the hyperplane in a transformed feature space, so that the classifier is non-linear in the original feature space, but linear in the transformed space.

An **advantage** of SVMs is that as the quadratic function is convex, SVM training finds a unique global minimum, whereas MLPs can suffer from multiple local minima. Other advantages are that SVM classifiers are simple to interpret geometrically, and can also be relatively flexible due to the kernel trick, which allows the SVM to approximate any computable functions.

A key **disadvantage** of SVMs is that choosing an appropriate kernel function is not a trivial task. A poor choice of kernel can sometimes lead to overfitting of the training data, as shown by Cawley and Talbot [9]. Furthermore, the quadratic optimization problem can potentially be computationally expensive in a high-dimensional space. One of the main **distinctions** between the two methods is that MLPs are trained on the minimization of empirical risk, while SVMs are based on the principle of maximum separation [10]. There is, for instance, no error propagation in SVMs to “train” the classification hyperplane.

4. Hypothesis

At a minimum, we expect both models to outperform the majority class weighting of 78.4%. As the dataset is relatively small, the advantages of the MLP model may not be readily apparent. With a relatively small dataset, we also expect the computation times for both models to be comparable.

5. Choice of training and methodology

We hold out 898 observations, or 18.3% of the original dataset, as the test set for model comparison, leaving 4000 observations (81.7%) for the training set.

Within the training set, we use 10-fold cross-validation and average the **classification accuracy** across the 10 training runs to choose the best parameters for SVM and MLP models. In searching for the best set of parameters for each model, we use Bayesian hyperparameter optimization for the SVM model and a grid search for the MLP. We then use the sets of parameters that yield the best classification accuracy to train the SVM and MLP models on the full training set of 4000 observations.

Finally, we use our trained SVM and MLP models to make predictions on the hold-out set for model evaluation. As evaluation criteria, we report traditional measures such as classification accuracy, precision, recall and F1-score, as well as the area under the receiver operating characteristic curve (AUC). Due to the slight class imbalance, if the various performance measures are in conflict, we prefer AUC as a model evaluation criterion.

6. Choice of parameters

6.1 Choice of parameters - MLP

For the MLP feed-forward network, we compared training accuracy and cross-validation error rate based on several different combinations of parameters and learning conditions.

The architecture of MLP networks is defined by the **number of layers** and neurons in those layers. If the hidden layer is too small, the backpropagation algorithm fails to converge to a minimum, but if there are too many nodes, it may result in the network overfitting the training data [7], so we tested up to 4 layers with 11, 8, 4 and 2 neurons in each layer respectively. We compared the use of various backpropagation **training algorithms**, such as Levenberg-Marquardt, (L-M) Bayesian regularization (BR), Scaled Conjugate Gradient (SCG), Gradient Descent (GDX) and Resilient Back-propagation (RB). Different sigmoidal **activation functions** such as *logsig* and *tansig* were also initiated.

The **learning rate** in backpropagation determines the step size taken during the iterative gradient descent learning process [6]. If this is too large, the network error will change erratically due to large weight changes, with the possibility of jumping over the global minima; conversely, if the learning rate is too small then training may take a long time. We thus used a combination of learning rates including 0.01, 0.1, 0.2 and 0.3. **Momentum** is used to support the gradient descent process if it becomes stuck in a local minimum [7]. We tested 0.7, 0.8, 0.9, and 1.0 as our momentum values.

We also implemented **early stopping**, with a minimum performance goal of 0.00001, and 10 validation checks. While a maximum number of epochs of 1000 were used, most training converged within 40 iterations. Where applicable, we tuned *mu*, the **Marquandt adjustment** parameter and the **regularization** parameter to reduce over-fitting.

Figure 4 shows the results of the hyper-parameters grid search. The selected MLP model and the set of parameters under which it was trained is highlighted in yellow. This model gave a combination of good overall training accuracy and a low validation error, and also had the advantage of having just two hidden layers. As it had fewer weights to learn, this could reduce the chances of overfitting, by adhering to the heuristic of Ockham's razor.

Learning algorithm	Activation fn	No. of neurons	Layers	Learning rate	Momentum	mu	Regularization	Training accuracy	Validation error
trainlm	logsig	11	1	0.1	0.9		0.5	85.1	18.2
trainbr	tansig	11	1					83.2	18.1
trainlm	logsig	11,8	2	0.1	0.9		0.5	83.6	17.9
trainlm	tansig	11,8	2	0.1	0.9		0.5	83.2	17.7
trainbr	tansig	11,8	2			0.005		85.0	17.6
trainbr	tansig	11,8	2			0.01		84.2	17.9
trainbr	tansig	11,8	2			0.001		84.2	17.7
trainbr	logsig	11,8	2			0.005		84.3	17.5
traingdx	logsig	11,8	2	0.1	0.9		0.5	79.9	21.4
traingdx	tansig	11,8	2	0.1	0.9		0.5	80.1	20.8
traingdx	logsig	11,8	2	0.01	0.7		0.5	79.5	20.3
traingdx	tansig	11,8	2	0.01	0.7		0.5	80.2	20.1
trainscg	logsig	11,8	2	0.1	0.9			81.2	19.4
trainscg	tansig	11,8	2	0.1	0.9			80.7	19.3
trainrp	tansig	11,8	2	0.1	0.9			80.9	19.2
trainrp	logsig	11,8	2	0.1	0.9			80.9	19.7
trainlm	logsig	11,8,4	3	0.1	0.9			83.5	17.8
trainlm	tansig	11,8,4	3	0.1	0.9			81.7	18.1
trainlm	logsig	11,8,4	3	0.1	0.9			87.2	18.4
trainlm	logsig	11,8,4	3	0.1	0.9			83.3	17.8
trainlm	logsig	11,8,4,2	4	0.1	0.9			84.1	18.5
trainlm	tansig	11,8,4,2	4	0.1	0.9			82.2	18.1
trainlm	logsig	11,8,4,2	4	0.1	0.9			85.9	17.8
trainbr	logsig	11,8,4,2	4			0.005	0.5	85.5	18.0
trainbr	tansig	11,8,4,2	4			0.005	0.5	84.7	18.0
trainlm	logsig	11,8,4,2	4	0.1	0.9		0.2	83.4	17.9

Fig. 4. Grid search of hyperparameters.

We found that the optimum number of neurons to use were 11 and 8 in a two-layered structure, with a *tansig* activation function, trained by Bayesian Regularisation. In general, the L-M and BR algorithms performed better than the SCG, RB or regular gradient descent algorithms. Having discovered the optimal structure, we sought to optimize around it, performing more iterations of the grid search in the vicinity.

6.2 Choice of parameters – SVM

We begin our hyperparameter search for the SVM model by choosing to optimize the **box constraint** and the **kernel scale** for three different **kernel functions**: linear, polynomial and Gaussian. In order to tune the hyperparameters, we use a Bayesian hyperparameter optimizer to evaluate the objective function, in this case the 10-fold cross-validated classification error. Bayesian optimization works by assuming a function mapping from the hyperparameters to the objective [6]. The Bayesian optimizer assumes a general prior distribution of this function mapping, and repeatedly updates this prior through iterative function evaluations.

We run the Bayesian optimizer for the three kernels with 10-fold cross validation, and 30 evaluations for each pair of hyperparameters. Figure 5 shows the estimated posterior distribution of the mapping after the 30 function evaluations for the linear, polynomial and Gaussian kernel respectively. Inspecting the shapes of the graphs can shed some information on the topography of the data. In particular, the failure of the linear kernel to achieve an error rate lower than 21.4%, as evidenced by the flat portion of Figure 5 (left), shows that the data is not linearly separable. In this case, the model achieves its best results by always predicting the majority class.

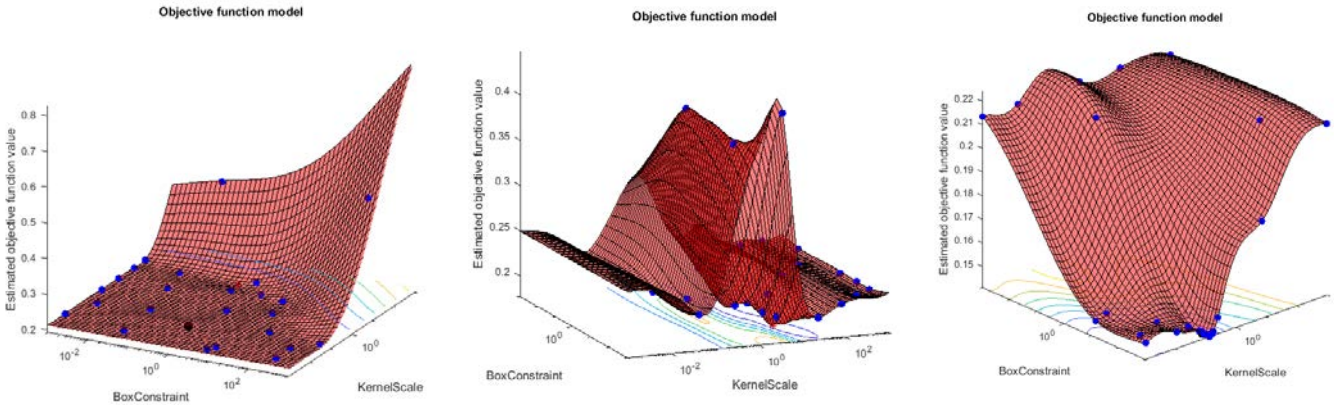

Fig 5. Cost function vs. box constraint, kernel scale for linear (left), polynomial (centre), Gaussian (right) kernels.

Figure 6 shows the summary of results of the optimization. As the **Gaussian kernel** achieves the lowest error rate of 14.1% with a box constraint of 560.62 and a kernel scale of 0.135, these parameters were chosen to train the SVM model.

Kernel	Optimal parameters		Error
	Box constraint	Kernel scale	
Linear	1.842	0.115	0.214
Polynomial	2.188	3.650	0.189
Gaussian	560.620	0.135	0.141

Fig. 6. Summary of Bayesian optimization results.

7. Experimental results and analysis

At this stage, we use the trained MLP and SVM models to make predictions on the test set, in order to evaluate how the models generalize to unseen data. Figure 7 shows the results of the models on the test set, with training set accuracy included for comparison. While the SVM outperforms the MLP on training and test **accuracy**, the MLP model outperforms on **recall**, **F1-score** and **AUC**.

	MLP	SVM
Training accuracy	85.0%	85.4%
Test accuracy	82.5%	84.5%
Recall	0.4854	0.3301
Precision	0.6623	0.9855
F1 score	0.5602	0.4946
AUC	0.8431	0.8146

Fig. 7. Summary of performance measures.

Both models show a decline in accuracy from the training set to the test set, a sign of **over-fitting**. The MLP model fares slightly worse in this regard, dropping in accuracy from 85.0% to 82.5%, while the SVM model drops from 85.4% to 84.5%. Computational time for training both models was comparable, with both taking about 2-3 seconds of wall clock time.

Although both models have a good accuracy, it is important to compare the **F1-score** to determine which model is better as accuracy on imbalanced data can be misleading. The F1-score uses the harmonic mean of recall and precision, providing a good balance between the two measure, and proving to be particularly useful with imbalanced datasets. It tends to favour classifiers that are strong in both precision and recall rather than classifiers that emphasize one at the cost of the other. Due to classifiers being more inclined towards the majority class, the F1-score also helps with identifying the members of the rare class, which makes it a satisfactory evaluation criterion in our project. The SVM had a test accuracy of 84.5%, with an F1-score of 49.5%, while the MLP had a test accuracy of 82.5% with an F1 score of 56.0%, making MLP the better model here. This can be explained further by examining the confusion matrices on the test set for both models (Figure 8), as well as the receiver operating characteristic (ROC) curves (Figure 9).

	Model predictions	
	MLP 1	**MLP 2**
Actual 1	641	51
Actual 2	106	100
	SVM 1	**SVM 2**
Actual 1	691	1
Actual 2	138	68

Fig. 8. Confusion matrices for MLP, SVM.

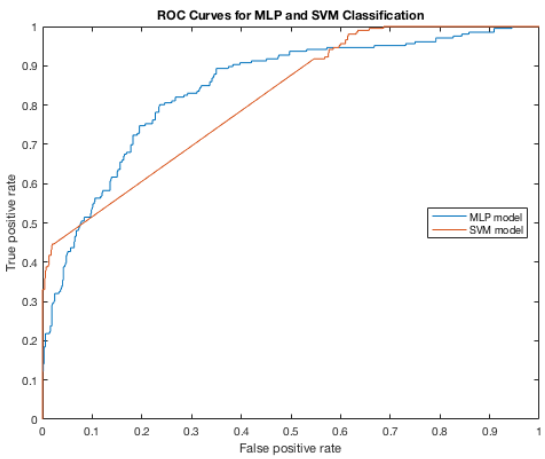

Fig. 9. ROC curve for MLP, SVM.

The confusion matrices reveal that the MLP correctly classified 641/692 as bad wine while misclassifying 51 as good wine. It also correctly classified 100/206 as good wine while misclassifying 106 as bad wine. In contrast, the SVM correctly classified 691/692 as bad wine while misclassifying 1 as good wine. It also correctly classified 68/206 as good wine while misclassifying 138 as bad wine. The SVM achieved a higher accuracy by predicting more of the majority class (bad wines) due to the imbalance in the data resulting in more false negatives and only one false positive, while the false negatives and

false positives are more evenly distributed in MLP. This is also evident in the ROC curve, where the SVM model performs better at a higher threshold due to the extremely low false positive rate, but the MLP model performs better at intermediate values.

Applying this to our use case of wine purchase, the SVM model results in the wine merchant missing out on good wines. Choosing which model is preferable is dependent on the use case. For instance, the wine merchant may want to avoid paying for bad wine (avoid false positives), in which case the SVM model is preferable. Or he may wish to collect more good wine, at the expense of buying bad wine, in which case the MLP model is preferred. Although the SVM is more accurate, we conclude the MLP is in fact the **more generalizable** model.

8. Conclusions, lessons learned and future work

In conclusion, although the MLP was the more generalizable model, a few limitations stood out. One disadvantage is the random initialization of weights by the backpropagation algorithm, which assigns random weight values to the neurons to avoid networks being stuck in the same local minimum each time they are trained. The random weights cause networks to be initialized differently in each training process, which can cause slight differences in the final results.

We found that the MLP training was not **robust** to cross-validation samples, varying 1-2% depending on how the folds were partitioned. One way to work around this would be to increase *k*, the number of partitions, although this can be computationally expensive.

For SVMs, the **choice of kernel** can be crucial, in this case improving accuracy by 7%. However, without any expert knowledge about the topology of the data in higher dimensions, it can be hard to make an educated guess to choose an appropriate kernel. The poorer generalizability of the SVM model shows that accuracy may not always be the best model evaluation criteria when training.

For future work, depending on the objectives and priorities of the wine merchant, we may add a penalty weight and create a different loss function to more heavily penalize false positives or false negatives as appropriate. As there was some imbalance in the dataset, resampling techniques such as SMOTE could also be applied to improve results.

9. References

[1] Cortez, P., Cerdeira, A., Alemeida, A., Matos, T. & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties, *Decision Support Systems*, 47, pp. 547-533.

[2] UCI Machine Learning Repository. 2018. Wine Quality Data Set. [ONLINE] Available at: https://archive.ics.uci.edu/ml/datasets/wine+quality. [Accessed 24 March 2018].

[3] Chawla, N.V., Bowyer, K.W., Hall, L. O. & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research*, 16, pp. 321-357.

[4] Zanaty, E. (2012). Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. *Egyptian Informatics Journal,* 13(3), pp.177-183.

[5] Byvatov, E., Fechner, U., Sadowski, J. and Schneider, G. (2004). Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug/Nondrug Classification*. ChemInform*, 35(5).

[6] Gardner, M. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14-15), pp.2627-2636.

[7] Attoh-Okine, N. (1999). Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance. *Advances in Engineering Software*, 30(4), pp.291-302.

[8] Cortes, C. & Vapnik V. (1995). Support-vector networks, *Machine Learning*, 20 (3), pp. 273-297.

[9] Cawley, G.C. & Talbot, N.L.C. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation, *Journal of Machine Learning Research*, 11, pp. 2079-2107.

[10] Barabine, N., Pallavicini, M., Petrolini, A., Pontil, M., and Verri, A. (1999) Support Vector Machines vs Multi-Layer Perceptron in Particle Identification, *ESANN'1999 proceedings – European Symposium on Artificial Neural Networks*, pp.257-262

[11] Snoek, J., Larochelle, H. & Adams, R.A. 2012. Practical Bayesian optimization of machine learning algorithms, *Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2, pp.2951-2959.

[12] Dreiseitl, S. and Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5-6), pp.352-359.