A Comparative Study in predicting wine quality with MLP and SVM

Fei Xie

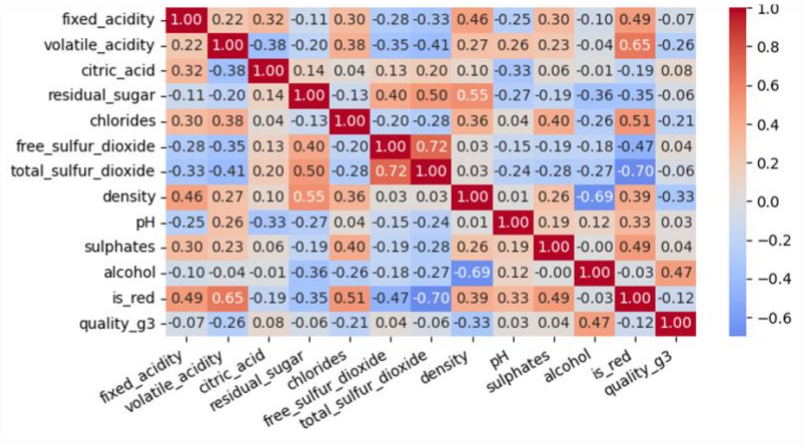
1. Problem statement and dataset

This study aims to perform supervised multi-class classification on wine quality dataset [1] using two neural computing algorithms, and then critically compare the results of the best model from each algorithm. The dataset consists of 6497 records, 12 feature columns and 1 label column. The dataset doesn't have missing value. The original 7 classes were regrouped into 3 classes due to extremely small sample size of some classes. After regrouping, imbalance classes problem still exists (percentages of three classes: 36.7%, 43.7% and 19.7%). Descriptive statistics and correlation analysis are shown in Table 1 and Figure 1. The two algorithms being evaluated are Multilayer Perceptron (MLPs) and Support Vector Machines (SVM).

Table 1. Features Summary

Features	Mean	Std	Min	Max
fixed_acidity	7.22	1.30	3.80	15.90
volatile_acidity	0.34	0.16	0.08	1.58
citric_acid	0.32	0.15	0.00	1.66
residual_sugar	5.44	4.76	0.60	65.80
chlorides	0.06	0.04	0.01	0.61
free_sulfur_dioxide	30.53	17.75	1.00	289.00
total_sulfur_dioxide	115.74	56.52	6.00	440.00
density	0.99	0.00	0.99	1.04
pH	3.22	0.16	2.72	4.01
sulphates	0.53	0.15	0.22	2.00
alcohol	10.49	1.19	8.00	14.90

Figure 1. Correlation Heatmap



2. Summary of two methods and hypothesis statements

MLP

MLP, also known as Artificial Neural Network, is a supervised algorithm that consist of multiple layers (input layer, output layer and multiple hidden layers) and multiple nodes within each layer. For MLP training, Backpropagation is the key to learn from data. During the training, data pass through the network layer by layer, with neurons taking and transforming input from previous layer using weights (usually randomly initiated in the beginning), bias and activation function, and then passing the output to the next layer until the final output is produced (Forward Propagation). Then the loss is calculated, and gradients of the loss function are taken with respect to each weight (Backpropagation). Once the gradients are taken, the weights are updated using gradient descent iteratively.

The first advantage of MLP is it can be applied to learn complex patterns and non-linear relationships in data due to its multilayer architecture and a wide choice of non-linear activation functions. Besides, it works well with large amount and high-dimensional data. However, MLP usually requires a large amount of data for training to achieve good performance, which can be

computationally expensive and time consuming, especially when the model architecture is deep and complex. Besides, MLP is sensitive to many hyperparameters such as number of neurons, learning rate, and initial weights. Choosing and tuning hyperparameters can be challenging and expensive. Another disadvantage is that MLP has a non-convex loss function and might thus plunge to a local minimum.

SVM

SVM is a another widely used supervised algorithm for classification. The idea of SVM is it tries to find the optimal hyperplane that best separates different classes in a dataset. SVM was originally designed for binary classification, but it can be extended to multi-class classification with the help of methods like one-against-one or one-against-all [2]. When dealing with non-linear separable problems, SVM uses kernel function map the input features into a higher-dimensional space where the classes can then be separable by a hyperplane.

The first advantage of SVM is it is effective in high dimensional spaces and still effective in cases where number of dimensions is greater than the number of samples [3]. Besides, it is more memory efficient as it uses only a subset of data points in the decision function (i.e., support vectors) [3]. Third advantage is that the loss function is convex, so it has only one minimum and thus avoids local minima issue. SVM has some disadvantages, for example, it is prone to overfitting when the number of features is much greater than the number of samples, but this is not the case for this study. Besides, SVM is not suitable for very large dataset. Another disadvantage is that SVM is sensitive to kernel and hyperparameters of kernel (e.g., gamma in RBF kernel [4]), so choosing kernels and hyperparameters is crucial.

The hypothesises for this study are:

a. SVM achieves slightly better accuracy than MLP on predicting wine quality rank.
b. SVM has a longer training time than MLP.

3. Methodology and implementation details

Training:

For model training, the preprocessed (e.g., one-hot encoding, regrouping labels) dataset were randomly split into training set (80%) and unseen test set (20%). The training set was used for model training, hyperparameter tuning and model selection, while the test set is used for final test of the best models of each algorithm. The whole process of MLP and SVM training used exact the same subsets to ensure fair comparison.

The training process consists of three steps for each algorithm. It started with fitting baseline models with simple architecture (e.g., 1 hidden layer for MLP) and default hyperparameters. Then a series of initial experiments were carried out to explore how specific input-level techniques, such as SMOTE oversampling and feature selection based on correlation analysis, will impact model. After initial experiments, the chosen type(s) of input, for example oversampled training data, will be used to perform model-level hyperparameter tuning and model selection. Due to computational resource and efficiency concern, random search was chosen for searching optimal combination of hyperparameters.

Evaluation:

This study used 5-fold cross-validation to evaluate models' performance and avoid data leakage and overfitting. The training set and validation set were split from the previous training set, the test remains unseen in this stage. Due to the nature of multi-class classification problem and the presence of imbalanced labels, this study used accuracy, precision, recall, F1 and confusion matrix as the quantitative metrics for measuring model performance in model selection and final test stage. Training time was also tracked as another aspect of measurement.

Parameters:

The choice of hyperparameters were based on two considerations: enhancing model capacity and preventing overfitting [5].

For MLP, number of hidden layers, number of hidden neurons, learning rate, momentum, activation function, and weight decay were chosen for tuning. Number of hidden layers, number of hidden neurons influence model's capacity to learn patterns from data, learning rate controls the step size in gradient descent. A too low learning rate leads to slower convergence and model might get stuck in local minimum. A too high learning rate might result in overshooting the optimal parameters and thus showing divergence or instability. The momentum helps SGD optimizer accumulate the gradient from previous steps and thus helps optimizer move out of local minimum. Activation function introduces non-linearity to the model and enhances model capacity by enabling it to learn complex patterns in data. Weight decay serves as a form of regularization and helps prevent overfitting. Early stopping was also used in MLP training for preventing overfitting. This study tried out three model depth: 1-hidden-layer, 2-hidden-layer, and 3-hidden-layer.

For SVM, kernel, 'C' and gamma were chosen for tuning. Kernel impacts the model's ability to capture complex patterns in the data as kernel functions transform the input data into a higher dimensional space in different ways. Tuning the kernel enables to find the most suitable transformation for specific input. 'C' is a form of regularization that controls the trade-off between maximizing the margin and minimizing the error. A small 'C' encourages a wider margin and thus helps prevent overfitting but may lead to more misclassifications. A large 'C' emphasis more on correctness but may lead to a too complex model and overfitting. Gamma impacts the shape of the decision boundary. A smaller gamma implies a larger influence radius, leading to smoother decision boundaries and potentially simpler models. A higher gamma makes the model more sensitive to individual data points, resulting in more complex model and a higher risk of overfitting.

Experiments results and model selection:

The results of initial experiments are shown in table 2. For MLP, oversampling and feature selection didn't produce significant difference in initial experiments, so all these three types of input (original, OS, and feature selection) would be tried out in the following tuning. For SVM, oversampling increased model capacity but feature selection didn't, so oversampled dataset with entire features was selected for further tuning.

Random search tuning results are shown in table 3 and table 4. For MLP, the results varies significantly when hyperparameter combination changes, especially when hidden neurons and momentum change. Increasing numbers of hidden layer from one to two enhanced accuracy, but accuracy didn't enhance significantly from 2-hidden-layer to 3-hidden-layer. Due to the general poorer performance of other kernels in initial trials, only 'rbf' kernel was selected for SVM's random search tuning. We noticed that SVM generated relatively stable results in several

rounds of ransom search tuning, and the combination of “large C with small gamma” didn’t produce significantly better results as some experiment shows [4]. The best models were selected according to validation accuracy and are highlighted in red colour and bold font.

Table 2. Baseline and initial experiments

Model	Val_acc	Run Time (s)
svm_baseline	0.6157	7.2536
svm_SMOTE	0.6340	10.3805
svm_sub	0.6051	10.1010
mlp_baseline	0.5688	2.1238
mlp_SMOTE	0.5657	1.6804
mlp_sub	0.5694	1.8931

Table 3. Random Search for SVM

Best model from	C	gamma	kernel	Val_acc	Run Time (s)
random search 1	1.0158	0.9193	rbf	0.7173	45.1699
random search 2	35568.3872	0.9612	rbf	0.7285	95.7993
random search 3	4737.8046	1.0585	rbf	0.7292	103.8188
random search 4	45.1432	0.7768	rbf	0.7297	57.9959
random search 5	20.1848	0.6157	rbf	0.7306	51.8797
random search 6	3.5177	0.7869	rbf	**0.7314**	50.7604

random search was run multiple time to try different hyperparameter space

Table 4. Random Search for MLP

Best model from RS	Momentum	Learning Rate Init	Hidden Layer Sizes	Alpha	Activation	Acc_Val	Run Time
best_model_1l	0.9689	0.0353	(16,)	0.0001	relu	0.5996	7.0662
best_model_1l_OS	0.9719	0.0696	(16,)	0.001	logistic	0.6158	7.1840
best_model_1l_OS_sub	0.9574	0.0925	(16,)	0.0001	tanh	0.5971	6.2223
best_model_2l	0.9164	0.0633	(60, 82)	0.001	relu	0.6102	17.3341
best_model_2l_OS	0.9677	0.0693	(15, 24)	0.0001	relu	0.6209	35.5531
best_model_2l_OS	0.9707	0.0527	(38, 44)	0.001	tanh	0.6321	17.4577
best_model_2l_OS	0.9561	0.0593	(51, 15)	0.001	tanh	0.6334	19.4699
best_model_2l_OS	0.9788	0.0592	(35, 67)	0.0001	tanh	0.6465	15.0583
best_model_2l_OS	0.9814	0.0287	(49, 36)	0.001	relu	0.6507	17.2505
best_model_2l_OS	0.9689	0.0295	(60, 82)	0.0001	relu	**0.6641**	11.7734
best_model_2l_OS_sub	0.9841	0.0764	(60, 82)	0.001	tanh	0.6295	26.2508
best_model_3l	0.9832	0.0220	(56, 70, 59)	0.001	tanh	0.6188	22.5849
best_model_3l_OS	0.9886	0.0202	(22, 79, 85)	0.0001	relu	0.6425	32.7669
best_model_3l_OS	0.9892	0.0093	(69, 19, 61)	0.0001	relu	0.6447	26.5133
best_model_3l_OS	0.9895	0.0323	(69, 19, 61)	0.001	tanh	0.6515	29.0657
best_model_3l_OS	0.9601	0.0558	(69, 19, 61)	0.0001	relu	0.6517	25.9370
best_model_3l_OS	0.9862	0.0229	(32, 35, 76)	0.0001	tanh	0.6535	29.2443
best_model_3l_OS	0.9736	0.0244	(56, 70, 59)	0.001	tanh	0.6622	33.8389
best_model_3l_OS_sub	0.9649	0.0472	(56, 70, 59)	0.0001	relu	0.6237	28.1180

random search was run multiple time to try different input types, hidden layers

4. Analysis and critical evaluation of results

Two best models were tested on unseen test set and evaluated by accuracy, precision, recall, F1 (see Table 5) and confusion matrix (see Figure 2). The colour scale indicates the magnitude of the metrics, and the arrows indicate the changes from validation results to test results.

Generalisation

Both MLP and SVM achieved similar level of generalisation when testing on unseen data (table 5). The overall accuracy, precision, F1 score decreased by around 4% in MLP and around 3% in SVM from validation set evaluation to test set testing. However, MLP shows slightly better generalisation on overall recall compared to SVM (decreased by 2% vs decreased by 3%). For generalisation in class-level, both models have poorer generalisation on class 2 (the minority class), with F1 score both decreased by 18%. However, both models achieved good generalisation on class 1 (majority class) and class 0.

Overall performance

Comparing overall accuracy, precision, recall, and F1 score between best MLP and best SVM (table 5), we observed a generally better performance of SVM than MLP, with around 8% higher scores across metrics. There is no significant difference among accuracy, precision, recall and F1 scores of each model’s results. This result indicates that SVM outperforms MLP in learning underlying patterns in this dataset. The slightly poorer performance and generalisation of MLP might have to do with the relatively small dataset and the overlapping of classes in this dataset.

To clarify, the precision, recall and F1 scores here were calculated using macro average because we have imbalance classes problem in dataset. Macro average is calculated with unweighted mean thus can penalise the model if it produces incorrect prediction in the minority class [6].

Class-level performance

Comparing the performance on predicting specific classes (table 5), we found that although both models achieved the best F1 score on predicting poor quality wine (class 0), they perform very differently on precision and recall metrics. MLP achieved good precision on class 0 (0.74) but poor precision on class 2 (0.49), and good recall on class 2 (0.77) but poor recall on class 0 (0.61). This means MLP is not that good at catching class 0 when class 0 is present, but when data is classified as class 0, many of them are correct. In contrast, MLP is good at catching class 2 when class 2 is present, but nearly half of those being classified as class 2 are incorrect.

The same trend was not found in SVM. SVM achieved a more balanced performance on predicting specific classes. It achieved good precision and recall on class 0 (0.77, 0.74 respectively), but relatively poorer precision on class 2 (0.63). Overall, SVM made less misclassification in predicting wine quality rank.

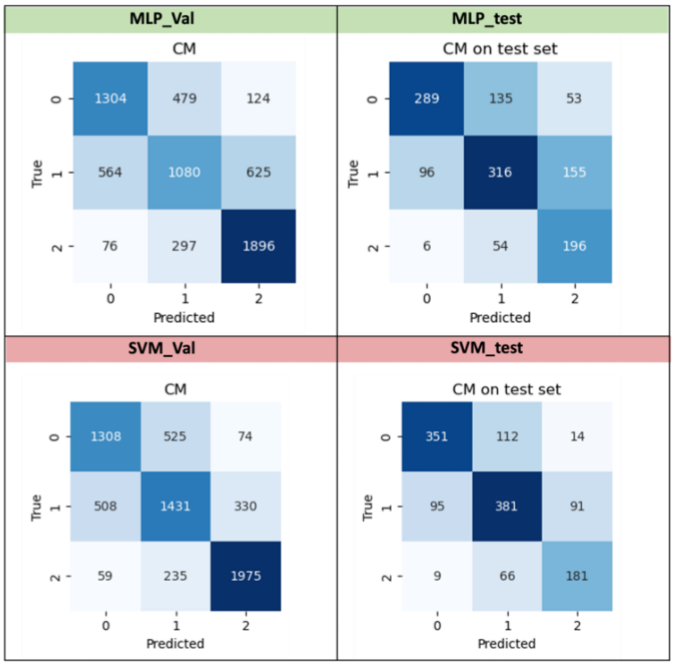
The less misclassification of SVM can be seen clearer from confusion matrix (Figure 1) as SVM has more predictions locating on diagonal. We can also observe from confusion matrix that MLP tends to misclassify class 1 as class 2 and misclassify class 0 as class 1. Noticeably, MLP has a much higher misclassification rate on the upper right corner, which means MLP distinguishes class 0 and class 2 poorer than SVM does.

Table 5. Performance of Best Models

	MLP Val	MLP test	SVM Val	SVM test
Overall Accuracy	0.66	0.62	0.73	0.70
Overall Precision	0.66	0.62	0.73	0.70
Overall Recall	0.66	0.64	0.73	0.70
Overall F1	0.66	0.62	0.73	0.70
Class0 Precision	0.67	0.74	0.70	0.77
Class1 Precision	0.58	0.63	0.65	0.68
Class2 Precision	0.72	0.49	0.83	0.63
Class0 Recall	0.68	0.61	0.69	0.74
Class1 Recall	0.48	0.56	0.63	0.67
Class2 Recall	0.84	0.77	0.87	0.71
Class0 F1	0.68	0.67	0.69	0.75
Class1 F1	0.52	0.59	0.64	0.68
Class2 F1	0.77	0.59	0.85	0.67
Train Time(s)	1.7507	-	3.0743	-
Test Time(s)	-	0.0023	-	0.8089

* Train Time: the duration of fitting best model on training data

Figure 2. Confusion Matrix



Training and testing time

Comparing the training time and test time, we found MLP (2-hidden-layer) has a significantly higher training speed than SVM (1.75 seconds, 3.07 seconds respectively), and when it comes to predicting on test set, this speed difference became even larger (0.0023 seconds, 0.8089 seconds respectively). The shorter training time of MLP might be due to the simple model architecture. The much shorter predicting time of MLP might be due to its computationally efficient forward propagation, which involves just matrix multiplications and activation functions, while SVM's prediction process still involves transforming input using kernel tricks.

To sum up, SVM is the preferable model in this study for three reasons. Firstly, it produces high overall accuracy, precision, recall, and F1 score. Secondly, it has a slightly better generalisation. Thirdly, it has a better and more stable performance in predicting specific classes, especially the minority class. Although SMV shows slower train and test speed, the slower speed is acceptable since the size of dataset is relatively small and the absolute speed difference is in an acceptable range (seconds). Besides, classifying wine quality is not a problem that requires absolute “real-time” low-latency prediction. If the task is some kinds of anomaly detection or making decisions and actions of self-driving cars where latency is not tolerant, the rapid prediction will be more crucial.

5. Lesson learned, future work and reference

This study trained and compared two neural computing algorithms on a wine quality classification problem. From this experiment I found SVM outperformed 2-hidden-layer MLP in terms of accuracy, precision, recall and F1 score. I also learned it takes longer time to train SVM and making prediction with trained SVM. Besides, hyperparameters such as C, kernel, gamma for SVM and number of hidden layers, number of hidden neurons, momentum, weight decay for MLP play a crucial role in enhancing model’s performance, as both models showed significant improvement after hyperparameter tuning. Thirdly, for imbalanced dataset, it is worthwhile to try balancing the classes because oversampled input generates better results. Finally, for multi-class classification problem, looking only at accuracy is not enough. When compared models with more specific metrics, such as looking at precision and recall scores on a per-class basis, I found more distinct difference in model performance.

Due to time and computational resource constraints, this study considered only random search techniques for hyperparameter tuning, future work can consider combining random search and grid search to explore more possible combinations of hyperparameter, especially for MLP, and further enhance model’s capacity and generalisation. Besides, the features in this study are not very discriminative among three classes, this might explain why the models didn’t reach very high scores after tuning. More feature engineering technique can be tried for further enhancing the models.

Reference

[1] ‘Wine Quality’. Accessed: Apr. 16, 2024. [Online]. Available: https://www.kaggle.com/datasets/rajyellow46/wine-quality

[2] C.-W. Hsu and C.-J. Lin, ‘A comparison of methods for multiclass support vector machines’, *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002, doi: 10.1109/72.991427.

[3] ‘1.4. Support Vector Machines’, scikit-learn. Accessed: Apr. 15, 2024. [Online]. Available: https://scikit-learn/stable/modules/svm.html

[4] ‘RBF SVM parameters’, scikit-learn. Accessed: Apr. 15, 2024. [Online]. Available: https://scikit-learn/stable/auto_examples/svm/plot_rbf_parameters.html

[5] ‘A Recipe for Training Neural Networks’. Accessed: Apr. 09, 2024. [Online]. Available: https://karpathy.github.io/2019/04/25/recipe/

[6] J. Mathew *et al.*, *A comparison of machine learning methods to classify radioactive elements using prompt-gamma-ray neutron activation data*. 2023. doi: 10.21203/rs.3.rs-2518432/v1.

Glossary and intermediate results

1. Glossary

MLP: multilayer perceptron

SVM: support vector machine

Baseline models: Models fitted with the simplest architecture or default hyperparameters. They served as a baseline to understand how much a model can learn without tuning.

Input-level tuning: Techniques implemented to dataset, such as oversampling, feature selection.

Model-level tuning: Techniques implemented to models, same as hyperparameter tuning in the context of this study.

2. Intermedia results and other implementations (MLP)

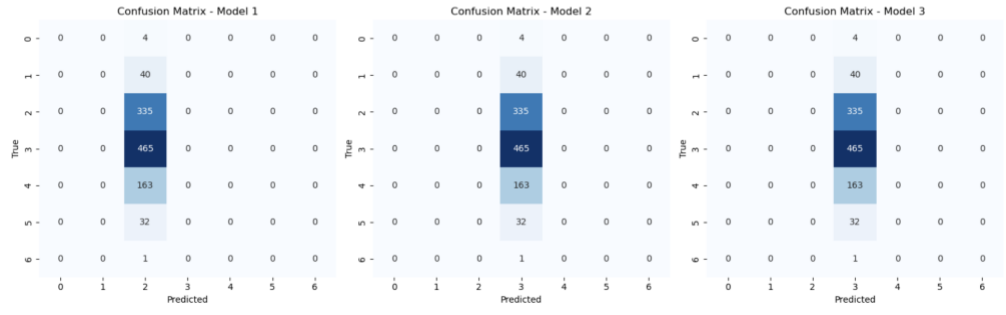
2.1. Poor results from initial implementations using Skorch:

epoch=100
Hidden neuron = 64 or (64,64) or (64,64,64)
Implementation 1: Adam, Weight decay (0.001), lr=0.01, dropout=0.3
Implementation 2: Adam, Weight decay (0.001), lr=0.01, dropout=0.3, early stopping, weight initialization,
Implementation 3: Adam, Weight decay (0.001), lr=0.01, dropout=0.3, early stopping, weight initialization, Batch Normalization

Accuracy	Implementation 1	Implementation 2	Implementation 3
1-hidden-layer	0.5856	0.5788	0.5779
2-hidden-layer	0.5760	0.5837	0.5885
3-hidden-layer	0.5865	0.5827	0.5760

2.2. Poor results when original classes were not regrouped.

```
Val:
Model 1 Acc: 0.3221
Model 2 Acc: 0.4471
Model 3 Acc: 0.4471
```



3. Intermedia results and other implementations (SVM)

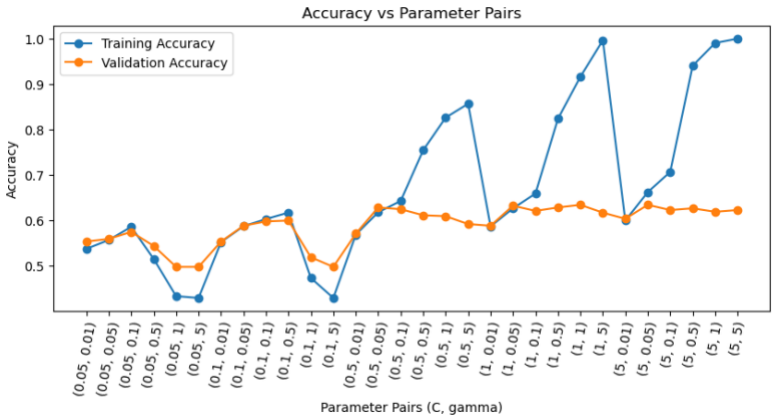
3.1. Experiments on kernels and scaling

From this experiments results: a. only rbf was chosen for tuning. b. still used standardization as scaling method.

Implementation	Accuracy	Train Time
rbf	0.6157	7.2536
sigmoid	0.4537	4.4401
linear	0.5617	6.5458
Poly (degree = 3)	0.5784	5.6553
Normalization	0.4298	14.4237
Polynomial	0.4160	25.5395

3.2. Grid search with poor choice of hyperparameter values.

Already overfitting on training set but accuracy on validation set still remains unimproved. This is one of the reasons why I switch to random search.



4. EDA for checking if classes are perfectly separable (no)