Modeling Wine Quality from Physicochemical Properties

Dale Angus
dangus@stanford.edu

Abstract

The purpose of this study is to find out if it is possible to predict what score a wine would be given based on its chemical properties and wine testers' opinion on the wine quality. I looked at a large dataset of both the red and white variants of Vinho Verde. Many different readily available classification algorithms were studied but the primary focus of this study is the binary and multi-class neural network classifiers that I created. To measure how close the predictions are, I used the top two predictions as basis for comparison with the actual rating. The results are very good and this opens up the possibility of assigning wine scores without the use of pricey professional wine testers.

1 Introduction

I started drinking wine about 15 years ago. I learned through the years that there are some wines that I enjoy more than the others. I also learned that what I deem as good wine has nothing or little to do with the price or the brand or the origin of the wine. I still find some expensive wines unpalatable. The same goes for some high-falutin brand names and also with some wines that originated from France or Argentina. In my attempt to avoid wasting money on unfinished bottles of wine during dinner or gatherings, I decided to purchase wine based on their scores.

I do know that these scores are purely subjective. "A wine rating is a score assigned by one or more wine critics to a wine tasted as a summary of that critic's evaluation of that wine. A wine rating is therefore a subjective quality score, typically of a numerical nature, given to a specific bottle of wine."

Nevertheless, my experience with this approach of buying wine based on high scores was pretty good. One limitation that comes with my approach is that this greatly limits my choice of wines to the scored ones. I wish to be able to avail of good wines from vineyards that have no access to these expensive wine critics.

So I am curious on how to score wines. An automated way, could there be?

I propose the development of YAWDA[1], the AI-powered wine-scoring system. The goal of YAWDA is to predict a wine rating that *closely matches* the one given by the humans.

2 Related Work

There are a few studies using the same dataset that can be found on the internet. The studies used include algorithms found in R or Scikit-Learn, and in one study a high performing fuzzy logic technique. The study by Cortez, Paulo et al. used regression with algorithms namely multiple regression, multilayer perceptron (a neural network), and support vector machines.. The study by Nebot, Angela et al. has a used a hybrid fuzzy logic techniques. The study by Vargas-Vera, Maria et al. used clustering and classification algorithms and includes an analysis of the input and output variables using the J48 algorithm. The study by Er, Yeşim & Atasoy, Ayten looked into binary classification between red and white wines using the Random Forest algorithm and studied multi-class classification using k-nearest neighborhood, random forests and support vector machines. All of these studies measured the performance of their models using the metric *accuracy*. [1][2][3][4]

[1] YAWDA – Yet Another Wine Dataset Analysis

3 Dataset and Features

The wine dataset used in this study was downloaded the UCI machine learning repository [7]. The wine covered in this dataset came from Minho, Portugal. There are two datasets related to the red and white variants of the Portuguese "Vinho Verde" wine. The data only includes eleven variables, ten physicochemical properties as inputs (see Table 1) and one sensory quality score as the output from 0 to 10. Data on grape types, wine brand and wine selling price are not included.

Table 1: The physicochemical data (input variables) and its corresponding statistics

Features	**Red wine**			**White wine**		
	Min	Max	Mean	Min	Max	Mean
Fixed acidity *g(tartaric acid)/dm*	4.6	15.9	8.32	3.8	14.2	6.855
Volatile acidity *g(acetic acid)/dm*	0.12	1.58	0.528	0.08	1.1	0.278
Citric acid *g/dm*	0.0	1.0	0.271	0.0	1.66	0.334
Residual sugar *g/dm*	0.9	15.5	2.539	0.6	65.8	6.391
Chlorides *g(sodium chloride)/dm*	0.12	0.611	0.087	0.009	0.346	0.046
Free sulfur dioxide *mg/dm*	1.0	72.0	15.875	2.0	289.0	35.308
Total sulfur dioxide *mg/dm*	6.0	289.0	46.478	9.0	440.0	138.361
Density *g/dm*	0.99	1.00369	0.997	0.98711	1.03898	0.994
pH	2.74	4.01	3.311	2.72	3.82	3.188
Sulphates *g(potassium sulphate)/dm*	0.33	2.0	0.658	0.22	1.08	0.489
Alcohol *%vol*	8.4	14.9	10.423	3.0	14.2	10.514

4 Methods

4.1 Scikit-Learn

Prior to developing the neural network model, a quick survey of the classification algorithms in Scikit-Learn was done. A complete list can be found in Table 6 below. Since Scikit-Learn does not include the preferred metric, Top-2 Categorical Accuracy, I wrote a custom metric so that the required performance data can be gathered for comparison purposes with the neural network model, YAWDA. Scikit-Learn was used to study feature selection and from there it was determined that all eleven features would produce the best prediction performance. This study did not include a deep dive into the nitty-gritty of the different algorithms used.

4.2 YAWDA, Neural Network Model

The majority of the study was focused on the development and understanding of neural networks. The code was developed in PyCharm using Python, and many software packages such as Anaconda, Numpy, Pandas, Keras, Scikit-Learn and Matplotlib. It was in the development of YAWDA that the importance of normalization was discovered. All experiments used the version of the dataset that has normalized input variables. The datasets were normalized using Scikit-Learn's StandardScaler function. The output variable, quality, was converted into a one-hot format as required by the softmax activated layer. Two models were developed, a multi-class classifier (YAWDA) and a binary classifier. The differences are highlighted in the table below.

Table 2: Model differences

	Multi-class Classifier	**Binary Classifier**
Loss function [9]	Categorical cross-entropy $-\sum_x p(x)\log(q(x))$	Binary cross-entropy *-(t log(sigmoid(o) + (1-t) log(1-sigmoid(o))*
Output activation [9]	Softmax $\exp x_{ij} / \sum_k \exp(x_{ik})$	Sigmoid $1/1 + \exp(-x)$
Performance Metric [10]	Top-2 Categorical Accuracy	Accuracy
Dependent variable	0 to 11 quality score (one-hot)	0 for white wine, 1 for red wine

4.2.1 Definitions

Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. A perfect model would have a log loss of 0. [11] As seen from the table, the formulae used are different when there are only two classes (*binary*) and more than two (*multi-class*).

Softmax function calculates the probabilities of each target class over all possible target classes (multi-class). *Sigmoid* is often used as the output layer's activation function for a binary classification. It takes a real number and returns the output value ranging from 0 to 1.

Top-2 Categorical Accuracy calculates the top-k categorical accuracy rate, i.e. success when the target class is within the top-k predictions provided. *Accuracy* simply calculates the accuracy rate for exact matches.

5 Experiments

5.1 Multi-Class Classification

For the initial experiments, the Adam optimizer was chosen as the default optimizer when studying the different aspects of the model such as regularization, number of layers, and various activation functions. After getting the models from these experiments, other optimizers were studied. Performance is measured using the metric *Top-2 categorical accuracy*. All experiments were run at least ten times and the highest accuracy is reported. The data was divided into train, validation and test using 56-24-20 split. The random state of the *train_test_split* [5] function was kept the same throughout.

5.1.1 Regularization

In the process of designing the neural network, several experiments encountered overfitting. The regularization technique for reducing overfitting, *Dropout*, was implemented. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network. In this study, I did not do an intensive study on the best dropout rate to use. Although, using 50% dropout for all hidden layers showed better results than a 25% dropout rate. The models were set to 50% dropout for all layers on all the experiments for studying the number of layers and activation functions.

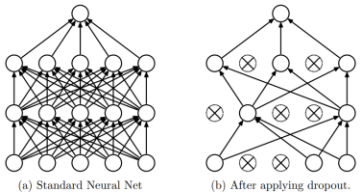

Figure 1: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. [6]

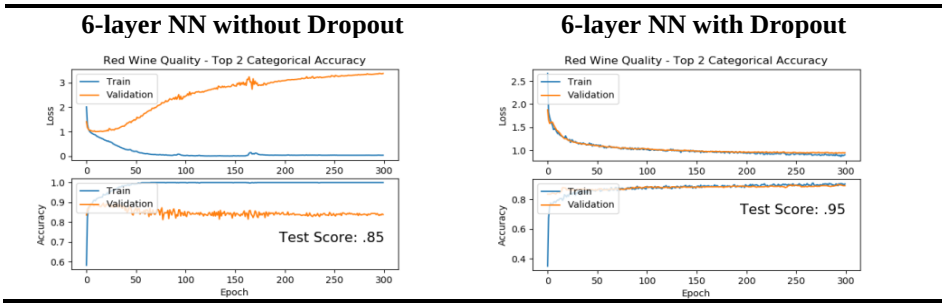

Figure 2: With and Without Dropout Regularization. Left: Shows the problem of overfitting and the low score. Right: No overfitting resulted in high score.

5.1.2 Number of Layers

Several experiments were performed to study what layer configuration would result in the best accuracy. The table below shows that the highest observed accuracy peaked to a certain number of layers and then declined from there. The model configuration for this experiment used all hidden layers set to 128 units, ReLU activation, and 50% Dropout rate.

Table 3: Searching for the best number of layers

Number of layers	Red wine	White wine
2 layers	0.9250	0.8735
3 layers	0.9313	0.8816
4 layers	0.9438	0.8816
5 layers	0.9469	**0.8888**
6 layers	**0.9500**	0.8795
7 layers	0.9500	0.8765
8 layers	0.9438	not tested
9 layers	0.9344	not tested

5.1.3 Activation

Several experiments were performed to see if a better accuracy using other activation functions in the hidden layers can be attained. ReLU turned out to be the superior activation function for the two models (red and white). The experiments for this used the best number of layers (see above) for the respective datasets, all hidden layers with 128 units, and 50% Dropout rate.

Table 4: Searching for the best activation function

Activation	Red wine	White wine
ReLU	**0.9500**	**0.8888**
Sigmoid	0.9500	0.8714

5.1.4 Optimizer

Several experiments were performed to see if a better accuracy from other activation functions can be attained.. Adam optimizer turned out to be the superior optimizer for the two models. The experiments for this used the best number of layers (see above) for the respective datasets, all hidden layers with 128 units, ReLU activation and 50% Dropout rate.

Table 5: Searching for the best optimizer

Optimizer	Red wine	White wine
Adam *lr=.001*	**0.9500**	**0.8888**
RMSprop *lr=.001*	0.9469	0.8755
SGD *lr=.001*	0.9281	0.8327
Adamax *lr=.005*	0.9438	0.8827

5.1.5 Multi-class Classifier Comparison

YAWDA managed to outperform the other classifiers that were tested during the Scikit-Learn experiments in the red wine dataset and came close to the best classifiers in the white wine dataset.

Table 6: Classifiers' Top-2 Categorical Accuracy

Classifier	Red wine	White wine
YAWDA (neural network)	**0.9500**	0.8888

Decision Tree Classifier	0.6500	0.6255
K-Neighbors Classifier	0.9188	0.8490
SVC	0.8500	0.8296
Logistic Regression	0.9250	0.8571
XGBoost Classifier	0.9344	0.8694
Extra Trees Classifier	0.9438	**0.9153**
Gradient Boosting Classifier	0.8875	0.8949
MLP Classifier (neural network)	0.9406	0.8857
Voting Classifier	0.9344	0.9112
Gaussian Process Classifier	0.9406	0.8714

5.2 Binary Classification

In this study, I combined the two datasets so I can study binary classification. I wanted to find out what combination of features will produce the highest performance in predicting whether a sample is a red or white wine. The metric used for binary classification is accuracy (not Top-2 categorical accuracy used in multi-class classification). I planned to run all possible combinations but after I saw that two features had already achieved 98% accuracy, I decided not to run all combinations anymore. Although I noted that some 4-feature combinations achieved 99% accuracy. The neural network model used was from the red wine multi-classification model with the output layer's activation changed from *softmax* to *sigmoid* and the loss function changed from *categorical cross-entropy* to *binary cross-entropy.*

The plot below shows that the combination of chlorides and total sulfur dioxide as the features that performed the highest accuracy compared to all other two-feature models. Adding either 1) fixed acidity and pH or 2) volatile acidity and free sulfur dioxide, to make a 4-feature model, achieved 99% accuracy.

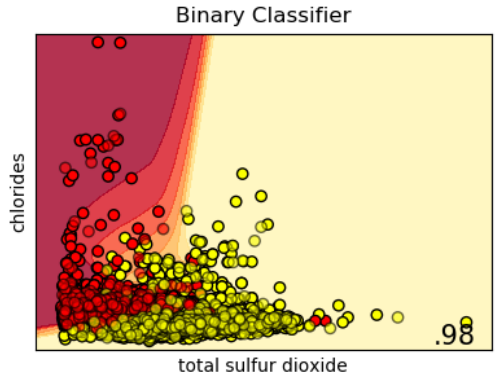

Figure 3: Scatter Plot for the best 2-feature model showing the contours of the prediction probabilities

6 Conclusion

In this study I studied if a neural network model can predict a wine rating that closely matches the actual rating. Arguably, the perfect metric for this is the Top-2 Categorical Accuracy. There were many aspects of the modeling process that had to be carefully studied and implemented. Normalization of the input variables helped improve the performance overall. Regularization was crucial in avoiding overfitting. Finding the right model parameters is iterative, time consuming but when done right results in remarkable outcomes.

I have not seen a similar dataset for quality classification where I can apply the multi-class classifier but I had a chance to apply the binary classification to one Kaggle dataset, MAGIC Gamma Telescope Dataset [12]. The goal is to classify high energy gamma particles in atmosphere. With little modification to the model, the output gave 87.33% accuracy classifying between signal or background particles.

7 References

[1] Maria Vargas-Vera, et al., (2017). A E-Business Case of Study: Modelling the Quality of the Wine using its Physicochemical and Qualitative Properties. International Journal of Knowledge Society Research

[2] Paulo Cortez, et al. (2009) Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems, Volume 47, Issue 4, November 2009

[3] Àngela Nebot, et al. (2015) Modeling Wine Preferences from Physicochemical Properties using Fuzzy Techniques. scitepress.org, 2015

[4] Er, Yeşim & Atasoy, Ayten. (2016). The Classification of White Wine and Red Wine According to Their Physicochemical Qualities. International Journal of Intelligent Systems and Applications in Engineering. 23-23. 10.18201/ijisae.265954.

[5] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[6] Srivastava, Nitish, et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15

[7] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

[8] Chen, Tianqi and Guestrin, Carlos. (2016) XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

[9] http://deeplearning.net/software/theano/library/tensor/nnet/nnet.html

[10] https://keras.io/metrics/

[11] https://www.kaggle.com/abhinand05/magic-gamma-telescope-dataset

8 Link to YAWDA Code

https://github.com/daleangus/proof_im_a_developer/blob/master/yawda.zip