Draft version December 16, 2019
Preprint typeset using LaTeX style emulateapj v. 12/16/11

PREDICTING WINE QUALITY SCORES USING MACHINE LEARNING

Bruce Chappell
email b.a.chappell@fys.uio.no
Francesco Anello
email f.anello1@campus.unimib.it/francean@student.matnat.uio.no

Draft version December 16, 2019

ABSTRACT

In this project we will study the chemical properties of wine and apply Neural Network, Decision Tree, and Random Forest Classifiers to these properties in and attempt to predict the quality score of the wine. We are interested in exploring these three classification methods and optimizing the hyperparameters of each to yield high accuracy predictive models. We found that optimized Random Forest and Neural Network Classifiers perform the best with Random Forests being the most optimal due to their fast runtime. These two methods resulted in accuracy scores of 0.64375 and 0.63438.

1. INTRODUCTION

Machine Learning techniques have been utilized in the academic world since the 1990's and are beginning to impact industries far from the typical computational disciplines. Wine making is an ancient process and in recent years scoring wine based on quality has become increasingly popular. Wines are assigned ordinal quality values based off of human taste test. Given that the fermenting of alcohol is, at its base, a chemistry problem, all wines have multiple chemical variables and a corresponding quality score assigned by human taste testers. Exploring which chemical variables contribute most to quality and developing a model to predict wine quality has potential to be a lucrative tool in the global wine business.

In this paper we will explore using different classification methods to predict wine quality based on 11 chemical variables. The methods we will use are the Multilayer Perceptron Classifier (we will use Neural Network for shorthand), Random Forest Classifiers, and Decision Tree Classifiers. We will then tune the hyperparameters for each model and achieve corresponding best fits. Finally we will compare the optimized models and make a judgement as to which method works best with this data.

The paper consists of 5 main following sections. In the theory section, we will briefly discuss the theory behind the classification methods used. In the data exploration section, we will explore the individual predictors and comment their relevance to the classification problem. In the results section, we provide the main results, focusing on accuracy scores and confusion matrices. We discuss the results in the discussion section, and provide closing thoughts in the conclusions section.

2. THEORY AND METHODS

2.1. *Decision tree*

The decision tree classifier is one of the simplest yet most extensively used techniques in multi-classification tasks. It involves solving a classification problem by asking a series of specific questions about the attributes of the test observation until a conclusion is reached about the class label for the specific instance. The set of questions and their respective answers can be graphically arranged in the form of a tree, which is a flowchart-like architecture consisting of nodes. The decision tree has three types of nodes connected by so-called branches:

- A root node, which is the source set. From this first node, it starts the splitting the data into subsets.
- Internal nodes. Both root and internal nodes test a specific attribute to separate records that have different characteristics.
- Leaf nodes or leaves, each of which do not split into further sub-nodes since it contains exactly one prediction (in this case a specific class label).

Hence, the leaf nodes contain the predictions we will make for new observations given to the trained model which has learned the structure of the training data. The decision tree model is trained by repeatedly dividing the target outcome variable along the values of the descriptive features using a measure of information gain. The tree finds rules (inputs and their levels) that best allow discriminating subjects of the training data in mutually exclusive terminal nodes defined by rectangular partitions of the x space as shown in Figure 1. Discrimination is in terms of the distribution of the target in such terminal nodes: maximum homogeneity of the target distribution within nodes and maximum heterogeneity of the target distribution between nodes. This method presents many advantages: classifying a

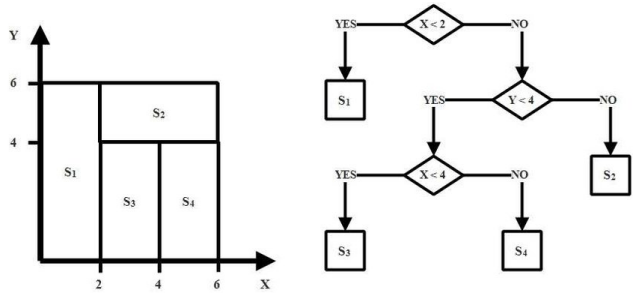

Fig. 1.— In the figure an example of how a simple decision tree is built. To be notice the tree boundaries are linear and axis-parallel.

new observation is extremely simple once a decision tree has been constructed and it is very easy to interpret and explain. Normally no reprocessing, like standardization, is necessary. In addition, it works with both numerical and categorical data and it allows us to use them both for classification and regression. Decision trees can fit nonlinear relationships, that means that it can do what a single line (like in linear regression) cannot. With regard to its simplicity, we can note some disadvantages: it normally does not present the same level of predictive accuracy as some of the other classification techniques. In addition, by using continuous features, the tree may become quite large and hence less interpretable. However, the major problem is that decision trees have a tendency to overfit the training data above all if the number of explanatory variables is relatively large and the number of records is relatively low. This problem can be resolved by implementing ensemble methods like bagging, boosting or random forests.

2.2. *Random Forests*

The Random Forest method consists of building a number of decision trees models for each bootstrap sample. When random forests starts to make a new split, it only considers a random sample of m predictors from the full set of p features. The subset size m is the (optional) tuning parameter. Random forests default to a subset size m that is the square root of the number of p predictors(note that if $m = p$, then Random Forest and bagging are the same). The Random Forest Algorithm starts drawing M bootstrap samples from original data. Then we grow several decision trees using "random" subsets of m predictors. This approach tends to perform very well compared to many other techniques and is robust against overfitting. In fact, considering a random sample of m predictors instead of all p predictors for splitting leads to random forests to "de-correlate" the bagged trees since it introduces noise. That is due to the fact that random choices of predictors, and thus different trees, leads to a greater reduction in variance. The latter is one of the most important properties of Random forests. In addition, it is robust to noise: outliers have less of an effect on individual models for the overall predicted values. So the main difference between a decision tree and a random forest is that the first one is built on the full data, using all the explanatory variables, whereas the second one randomly select a limited number of observations and features to build different decision trees from and then averages the outcomes.

2.3. *Neural network*

The theory behind Neural Networks was heavily treated in Project 2, which can be found in the Github repository listed in [7]. So as to not present a boring rephrasing of our previous work, we encourage readers to look there for a summary of the theory behind neural networks. Our main focus regarding neural networks in this project revolves around tuning hyperparameters; namely, the learning rate, the back propagation solving method, the activation function, and the number of hidden layers and their respective sizes.

3. DATA EXPLORATION

The wine data comes from Minho, a northern region of Portugal and it was collected from May 2004 to February 2007. The data set is available from the UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science. It contains 1599 red wines with 11 measurable physicochemical properties and quality as outcome variable. It has been proposed for both, regression and classification, by Cortez (2009). In this paper the dataset is viewed as a multi-classification problem. The response variable is an integer in the range 0 (very bad) to 10 (very excellent). For the purpose of our study we treat the quality output variable as an ordinal variable. All input variables are continuous and their main descriptive statistics are summarised in Table 9 in the appendix. At first glance, it seems likely that a prior cleaning has been made on this dataset since there are neither visible abnormalities nor missing values for any of the explanatory variables. Even if the variables assume values in the same order of magnitude, standardizing is here necessary since the variables are measured on different scales thus making it easier to compare and use them together. In regards to the outcome variable, the plot below shows that its range is between 3 to 8 and the classes are not balanced since the majority of wines fall into the medium quality range, with few in the high and low quality range. In order to understand which

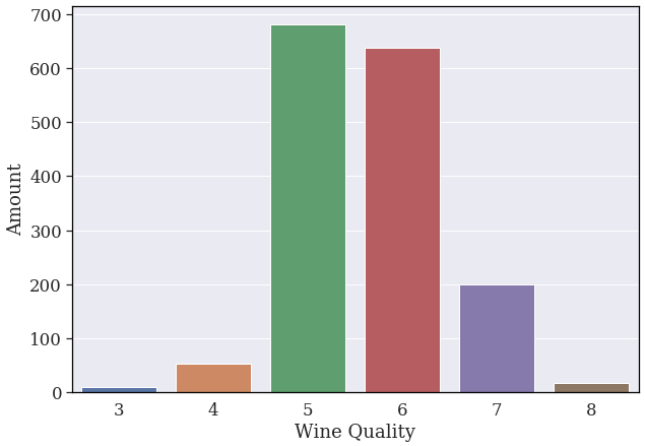

FIG. 2.— Frequency distribution histogram of the outcome variable

chemical properties influence the quality of red wine, it is interesting to analyse the correlation matrix. As shown in figure 3, the larger correlation coefficients (in absolute value) between the outcome variable and the explanatory ones are alcohol (0.48) and volatile acidity (-0.39). The other features are also correlated to the response variable but to a lesser degree. Only the variables residual sugar, free sulfur dioxide and pH do not present a correlation coefficient significantly different than 0. It is also interesting to study how the features are correlated among us. A lot of independent variables are correlated among themselves, for example: fixed acidity is highly correlated to density and citric acid (0.67), citric acid to volatile acidity(0.39) and pH (-0.54), alcohol to density(-0.5). Since the variables present significant correlations among themselves, it is convenient to consider a specific statistical measure called partial correlation coefficient:

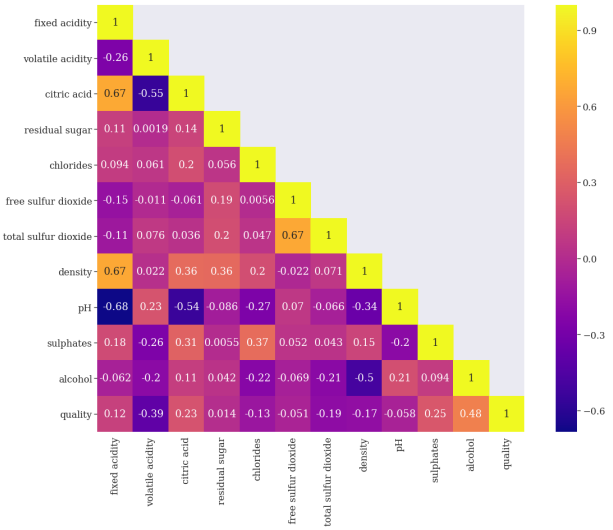

FIG. 3.— Correlation matrix for the physiochemical properties of the wine.

it describes the strength of a linear relationship between two variables, holding constant the other variables. This allow us to measure the pure strength of the linear relationship between two variables (in particular between the outcome and the explanatory variables) after setting for relationships implicating all the other variables. The partial correlation matrix is shown in the figure below: Here, it is immediately noticeable

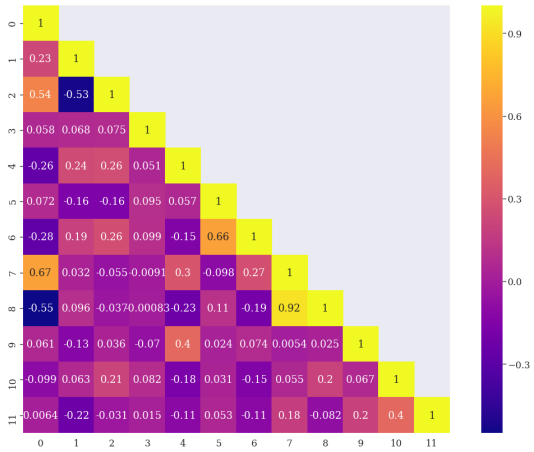

FIG. 4.— Partial correlation matrix

how the correlation coefficient between fixed acidity and the outcome variable collapses, changing from 0.12 to 0.0064. This is justified by the fact that it is highly correlated to other explanatory variables. The same goes for the variable citric acid. For the other features no important differences are highlighted. Hence, we notice that relevant features for the prediction of our outcome rating are volatile acidity, chlorides, total sulfur dioxide, density, sulphates and alcohol. For these main variables, plots are made in order to summarize the central tendency, dispersion and shape of each single variable. In particular box plots are used in order to facilitate comparisons across levels of the outcome variable, which is categorical, with respect to an explanatory variable.

Generally we notice that considerable dispersion

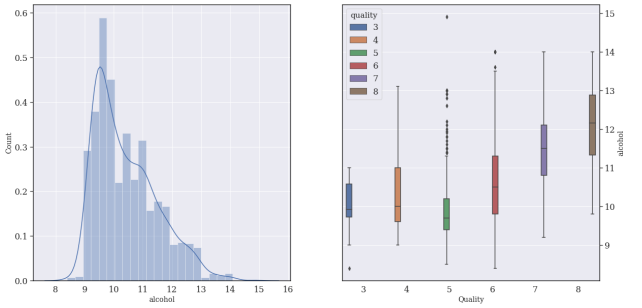

FIG. 5.— Frequency distribution histogram of *alcohol* and its box plots with respect to quality categories

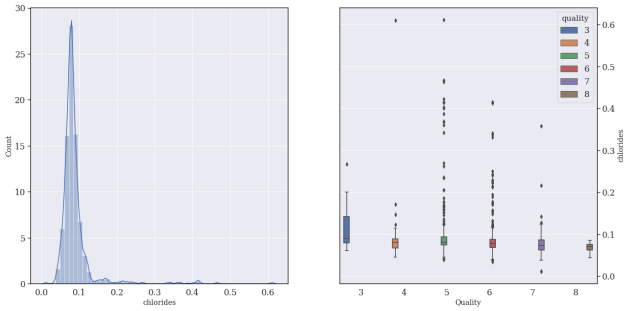

FIG. 6.— Frequency distribution histogram of *chlorides* and its box plots with respect to quality categories

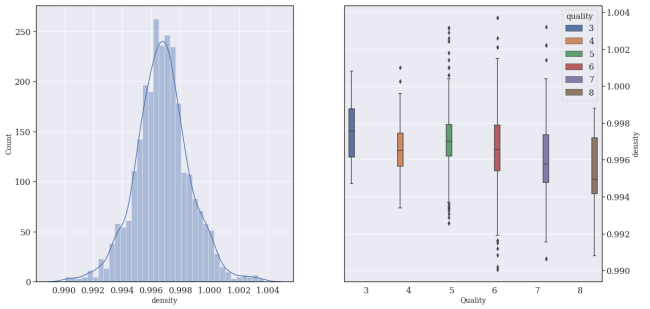

FIG. 7.— Frequency distribution histogram of *density* and its box plots with respect to quality categories

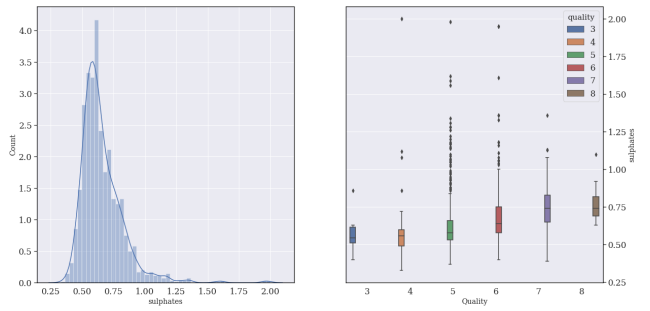

FIG. 8.— Frequency distribution histogram of *sulphates* and its box plots with respect to quality categories

and several outliers are present for these variables, particularly in the middle central categories. Taking into account the small scale of the dataset and the plausibility of the outliers, we do not exclude them from our study. Analysing the box plots we can note that, generally speaking, the greater the volatile acidity the lower the quality of the wine. On the contrary, the higher the alcohol percentage or the quantity of sulphates, the

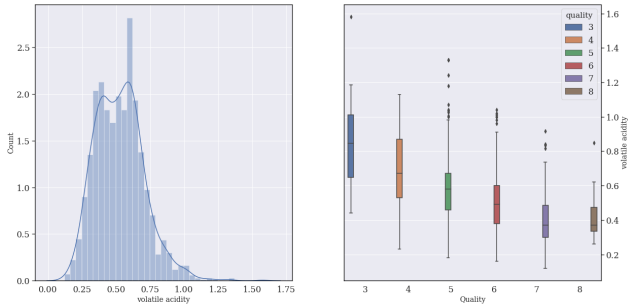

FIG. 9.— Frequency distribution histogram of *volatile acidity* and its box plots with respect to quality categories

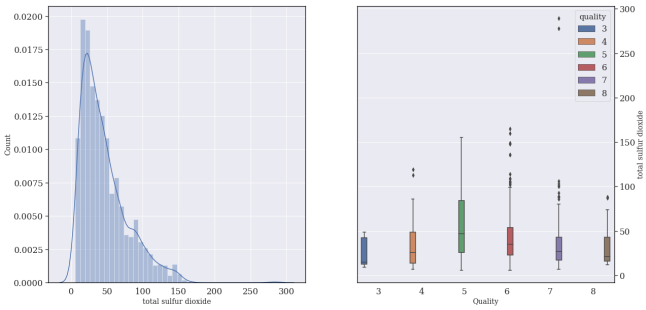

FIG. 10.— Frequency distribution histogram of *total sulfur dioxide* and its box plots with respect to quality categories

better the ranking. For the other variables, no specific patterns are remarkable from these plots. Those results are in perfect agreement on the previous correlation results. Regarding the distribution of the selected variables, we can see that the density and volatile acidity variables are the only ones that present approximately a normal distribution. The other ones are skewed to the right. It is interesting to see if it is possible to reduce the dimensionality of the problem using Principal Component Analysis (PCA) in order to simplify it and still achieve a good result. For the standardized training data, the following scree plot is observable :

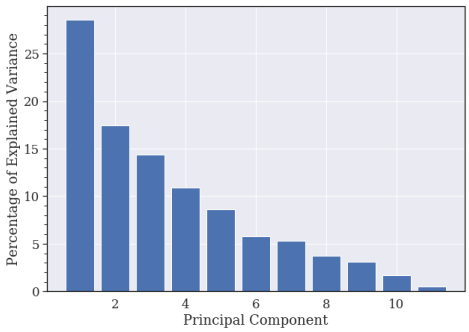

FIG. 11.— This plot reports the percentage of reproduced variance by each principal component(in ordinate) with the respect to its order number (in abscissa). The PC before the "elbow" are normally considered for analysis but in this case this method results useless since no "elbow" is remarkable.

In general, at least the first two PC are kept in study. They explain respectively 28.6 % and 17.4% of the total variance, for a total of 46%. To understand the meaning of these two PC, it is important to consider how much each explanatory variable contributes to the construction of each PC. This is possible considering the loading scores, reported in the table below:

TABLE 1
LOADING SCORES FOR EACH EXPLANATORY VARIABLE FOR THE FIRST TWO PC

Variable	*PC1*	*PC2*
Fixed acidity	**0.486880**	-0.103140
Citric acid	**0.460336**	-0.147740
pH	**-0.438818**	<0.01
Density	**0.395736**	0.234375
Sulphates	0.251823	-0.044525
Volatile acidity	-0.229951	0.271191
Chlorides	0.219922	0.143661
Residual sugar	0.144961	0.268519
Alcohol	-0.121612	**-0.376981**
Free sulfur dioxide	-0.035465	**0.521512**
Total sulfur dioxide	<0.01	**0.574514**

We can see that for the first PC the most important variables are Fixed acidity, Citric acid, pH and Density while for the second one alcohol, free and total sulfur dioxide are most important. Thus, the two PC represent different physicochemical aspects. From 12 we can see that the two first PC do not manage to discriminate and find differences for the different quality rankings. Hence, Principal Component Analysis does not seem to be relevant for this study.

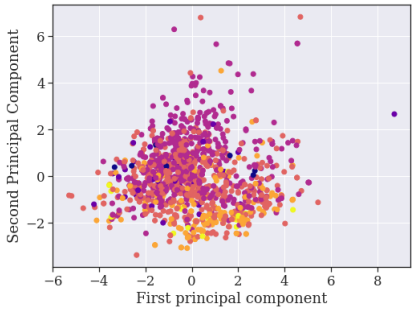

FIG. 12.— Individual reduced-space plot generated by the first two PC components

4. RESULTS

In this section all results are obtained using SciKit-Learn's MLPClassifier, RandomForestClassifier, and DecisionTreeClassifier methods. To begin, multi dimensional set of hyperparameters was defined for each model. Then Sk-learn's RandomizedSearchCV function was used to solve the model for 50 random samples of the defined hyperparameters. We then used a more rigid GridSearchCV around the optimal parameters found from the RandomizedSearchCV. Both of these functions utilize cross validation when solving the models using the different hyperparameters. This process allowed us to explore a wide range of hyperparameter options while also saving on computing time.

As an initial test, we applied the three trained models on the training data with their default settings yielding the following results.

TABLE 2
Baseline scores for the SK-learn methods without tuning any parameters.

Model Type	Accuracy Score
RandomForestClassifier()	0.54375
DecisionTreeClassifier()	0.54730
MLPClassifier()	0.54375

We will now tune the hyperparameters of each method in an attempt to improve the accuracy score. It should be noted that simply guessing all 5's for quality would result in an accuracy score of 0.422987.

4.1. *Random Forest Classification*

For the random forest classifier we began with a random grid search over the following hyperparamter values

TABLE 3
Initial random grid search for the Random Forest Classifier

Hyperparameter	Values
n_estimators	100, 200, 300, . . . 2000
max_features	'log2', 'sqrt'
max_depth	10, 20, 30, . . . 110
min_samples _split	2, 5, 10
min_samples_leaf	1, 2, 4
bootstrap	True, False

Through our randomized grid search, we found that the best parameter from the random grid search were

TABLE 4
Best parameters from Random Forest random grid search.

Hyperparameter	Values
n_estimators	300
max_features	'log2'
max_depth	80
min_samples _split	2
min_samples_leaf	2
bootstrap	False

which resulted in an accuracy score of 0.5746677. We then fix all hyperparameters except n_estimators and max_depth and expand around these resulting in the following plot

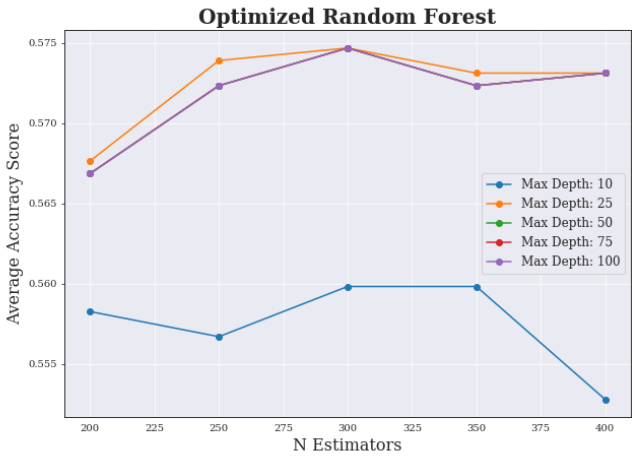

Fig. 13.— Hyperparameter comparison for Random Forest Model. Notice max_depth 50 and 75 were plotted over exactly by other model results.

Here we obtain an optimal model using $n\_estimators = 300$ and $max\_depth = 25$ with an accuracy score of 0.5746678. Using this optimized model with the test data we obtain an optimal accuracy score of 0.64375 and the following confusion matrix.

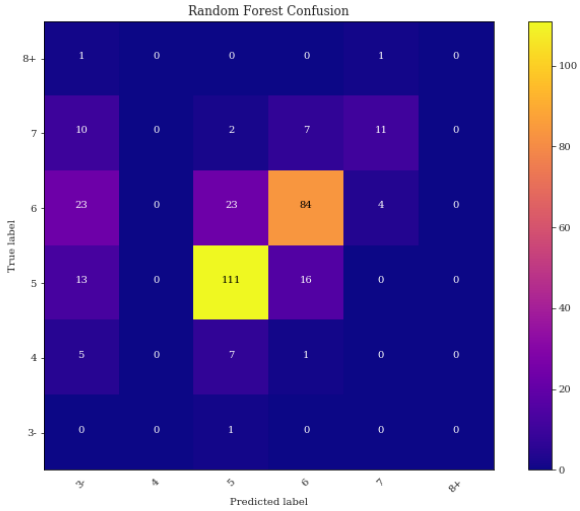

Fig. 14.— Confusion Matrix for the optimized Random Forest model.

4.2. *Neural Network Classification*

For the neural network classifier we began with a random grid search over the following hyperparamter values

TABLE 5
Initial random grid search for the Neural Network.

Hyperparameter	Values
learning_rate_init	1×10^{-3}, 1×10^{-1}
hidden_layer _sizes	(1000,)
hidden_layer _sizes	(1000,1000)
hidden_layer _sizes	(1000,1000,1000)
hidden_layer _sizes	(1000,1000,1000,1000)
solver	'adam', 'sgd', 'lbfgs'
activation	'relu', 'logistic'

The initial random grid search suggested the best paramaters were

TABLE 6
Best parameters from the Neural Network random grid search.

Hyperparameter	Values
learning_rate_init	1×10^{-1}
hidden_layer _sizes	(1000,1000,1000)
solver	'sgd'
activation	'relu'

with an accuracy score of 0.584050. Next we take the best random hyperparameters and expand around the learning rate and hidden layer sizes resulting in the following plot

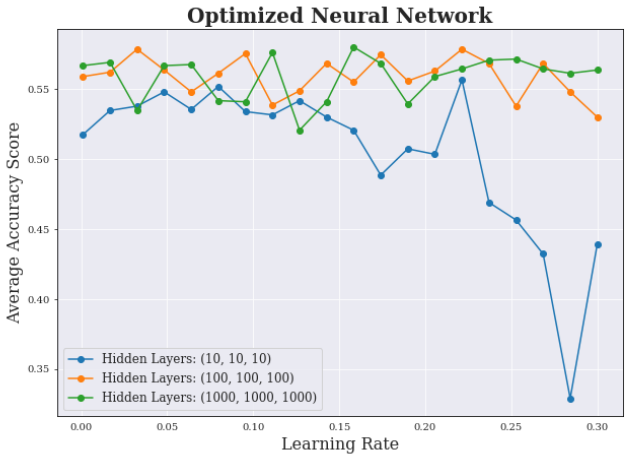

Fig. 15.— Hyperparameter comparison for the optimized neural network. Notice that (100,100,100) and (1000,1000,1000) exhibit similar behavior.

We then train the optimized neural network with (1000,1000,1000) layers and a learning rate of 0.158368 using the training data. Using this model on the test data results in an accuracy score of 0.634375 and the following confusion matrix.

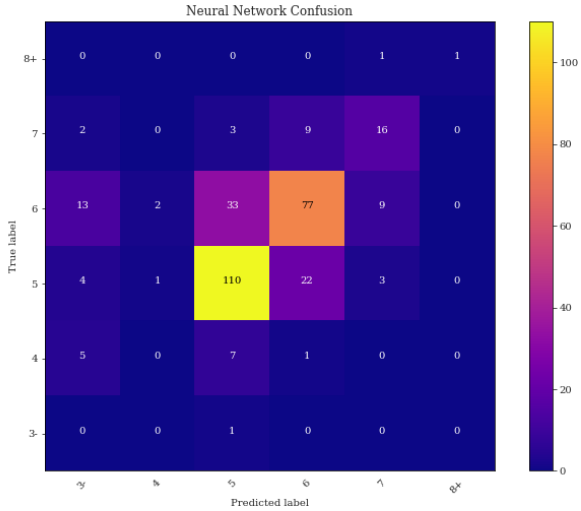

Fig. 16.— Confusion matrix for the optimized Neural Network

4.3. *Decision Tree Classifier*

Again, we begin with a random grid search over the following hyperparameters

TABLE 7
Initial random grid search values for the Decision Tree.

Hyperparameter	Values
criterion	'gini', 'entropy'
max_depth	1, 2, 3 . . . 10
max_features	1, 2, 3 . . . 11
max_leaf_node	2, 5, 10, 15
min_samples_split	2, 4, 6, . . . 50

This search suggested the best parameters were the following:

TABLE 8
Initial optimal hyperparameters for the Decision Tree from the random grid search.

Hyperparameter	Values
criterion	'gini'
max_depth	7
max_features	9
max_leaf_nodes	15
min_samples_split	40

These hyperparameters gave an initial accuracy score of 0.547302. We now fix all hyperparamters except max_depth and max_leaf_nodes which results in the following plot.

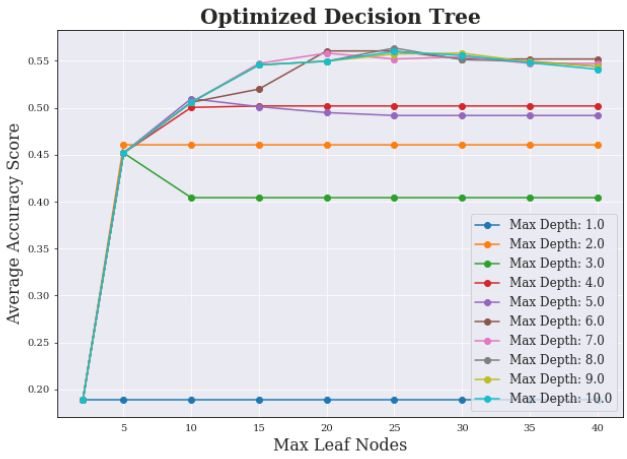

FIG. 17.— Hyperparameter comparison for optimized Decision Tree model.

We find the optimal decision tree to have a $max\_depth = 8$ and $max\_leaf\_nodes = 25$. Training this model with the training data and fitting to the test data we get an accuracy score of 0.578125 and the following confusion matrix.

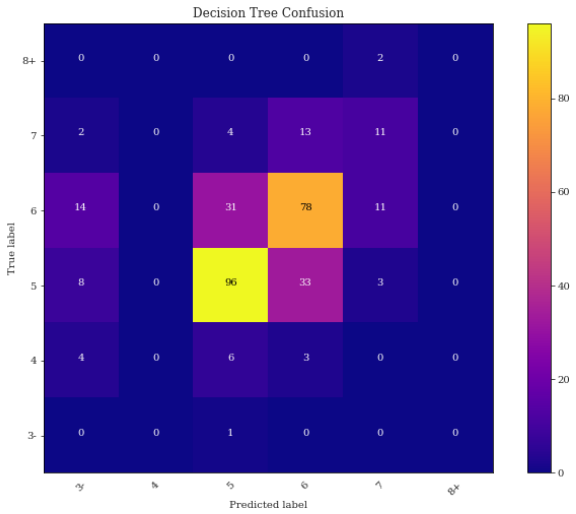

FIG. 18.— Confusion matrix for optimized decision tree.

5. DISCUSSION

To begin our discussion, we will preface by noting that quality scores 5 and 6 make up 81.70% of our data. Through a random grid search and a rigid grid search of hyperparameters, we were able to see an accuracy score improvement for the Random Forest Classifier from 0.54375 to 0.64375. In Fig 12 we observe some slightly odd behavior as it appears that four of the five models behavior nearly exactly the same. In Fig 13 we see that the model does the best job predicting qualities equal to 5 and 6 and struggles the most with predicting quality equal to 3. The optimized neural network saw an accuracy score improvement from 0.54375 to 0.63438. Interestingly, Fig 14 shows that when increasing hidden layer sizes from (100,100,100) to (1000,1000,1000) there is essentially no improvement in accuracy score. The confusion matrix in Fig 15 shows similar behavior to what was observed for the random forest. The model does well predicting 5's and 6's and struggles with the lower quality scores. Continuing on to the Decision Tree, by tuning hyperparameters we were able to increase our accuracy score from 0.54730 to 0.578125. This is a notably worse increase in accuracy score that the previous two methods. In Fig 16 we see that the accuracy score stabilizes around 55% for tree depths between 6 and 10. The decision tree confusion matrix in Fig 17 has lower values on the bottom left top right diagonal showing this model struggled to accurately predict the quality score of the inputs.

6. CONCLUSIONS

In our analysis, we have shown that for this data set, the MLPClassifier and RandomForestClassifier functions predict the wine quality data set the best when the hyperparameters are tuned accordingly. These two methods resulted in accuracy scores of 0.64375 and 0.63438 when trained on the training data and then predicting the test data. The DecisionTreeClassifier function proved to be less successful with an optimal accuracy score of 0.578125. One interesting result was that increasing neural network complexity did not necessarily end in better results. When it comes to deciding between the neural network and the random forest for this model, the decision comes down to run time. The neural network took 23.25 seconds to run where the random forest only took 1.25 seconds. Based off this criteria, with similar accuracy scores, we conclude that a random forest classifier is the best model to predict the target values of the data.

6.1. *Further research*

To better our results, we could benefit from having a larger data set to run experiments on as the set we used only had 1599 instances. Despite our best efforts, we could not manage to find a model that resulted in a satisfactory prediction of the data. Going forward we could explore other classification methods or work more closely with fine tuning the methods we used.

REFERENCES

[1]Hjorth-Jensen, Morten Lectures notes in FYS-STK4155.Data analysis and machine learning: https://github.com/CompPhysics/MachineLearning
[2]Pankaj Mehta, A high-bias, low-variance introduction to Machine Learning for physicists, May 29, 2019
[3]Trevor Hastie, Robert Tibshirani, and JH Friedman. The elements of statistical learning: data mining, inference, and prediction. 2009.
[4]Piccolo Domenico, Statistica, 21 Oct 2010.
[5]Azzalini, Scarpa. Data Analysis and Data Mining: an introduction, 2012-04-23.
[6]Zani S., Cerioli A. Analisi dei dati e data mining per le decisioni aziendali, Giuffrè Editore, Milano, 2007.
[7]Pang-Ning Tan, Michael Steinbach, Vipin Kumar. Introduction to data mining. Pearson education, 2006.

7. APPENDIX

All source code, data and figures can be found at the github repository: https://github.com/bruce-chappell/FYS4155/tree/master/project3

TABLE 9
Descriptive statistics of explanatory variables

	fixed acidity	*volatile acidity*	*citric acid*	*residual sugar*	*chlorides*	*free sulfur dioxide*	*total sulfur dioxide*	*density*	*pH*	*sulphates*	*alcohol*
mean	8.320	0.528	0.271	2.539	0.087	15.875	46.468	0.997	3.311	0.658	10.423
std	1.741	0.179	0.195	1.410	0.047	10.460	32.895	0.002	0.154	0.170	1.066
min	4.600	0.120	0.000	0.900	0.012	1.000	6.000	0.990	2.740	0.330	8.400
25%	7.100	0.390	0.090	1.900	0.070	7.000	22.000	0.996	3.210	0.550	9.500
50%	7.900	0.520	0.260	2.200	0.079	14.000	38.000	0.997	3.310	0.620	10.200
75%	9.200	0.640	0.420	2.600	0.090	21.000	62.000	0.998	3.400	0.730	11.100
max	15.900	1.580	1.000	15.500	0.611	72.000	289.000	1.004	4.010	2.000	14.900

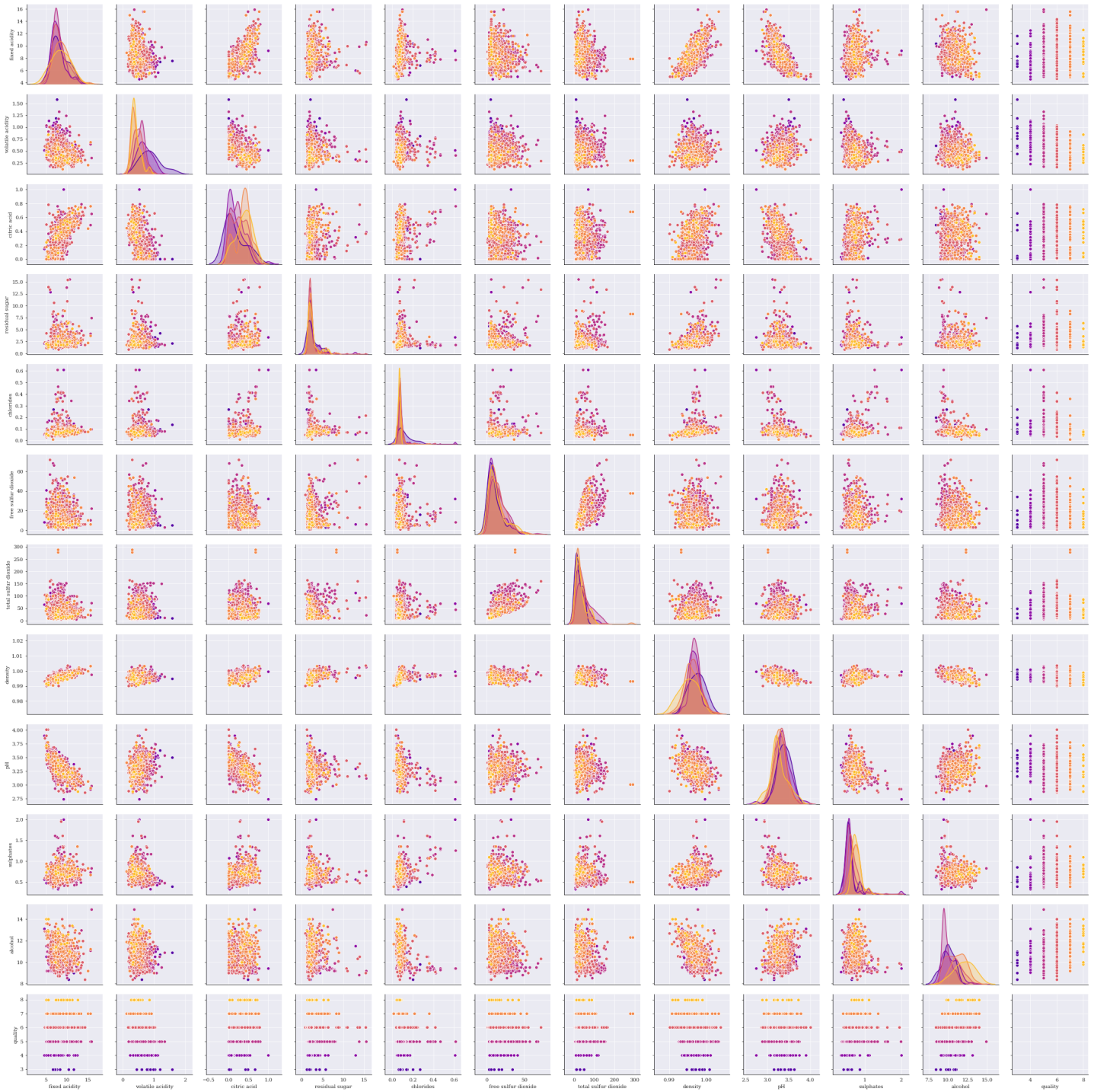

FIG. 19.— Scatterplot representing pairwise relationships in red wine dataset. In the diagonal, plots showing the univariate distribution of the data for a specific variable in its specific column are drawn.