Analysis on classifying red wine quality

Li Hanlong

November 23, 2023

Abstract

As machine learning has now become a powerful tool in solving multiple real life questions. We wish to introduce this technique to classify the quality of the red wine as well. This project aims to use machine learning techniques, like logistic regression, random forest, to help red wine manufacturers using the 11 chemical features to label the quality of the redwine, and further achieve profit maximization. In chapter one, we first introduce the background of this problem. Giving a brief introduction of the dataset we use and corresponding pre-processing method. Based on the nature of the target variable, we choose to use multi-classification, and define our objective in this problem. After that, we put forward our error metrics based on the objective and introduce the package we used during training. In chapter 2, we illustrate the models we have used, including logistic regression, random forest, XGboost. In chapter 3, we conclude the final result and put forward the final model of our project, followed by a discussion of future work in chapter5.

1 Introduction

The 'Red Wine Quality' dataset is a collection of attributes and quality ratings for red wines.The dataset comprises 11 physicochemical features, including acidity levels, alcohol content, and volatile acidity, along with a quality rating assigned by human evaluators. The quality ratings range from 3 to 8, where higher values correspond to better perceived quality.

To explore variable importance of 11 physicochemical features and to find important features, we conduct principal component analysis (PCA). We figure out PCA is not effective because we need to include 7 principle components to achieve a total variation explained larger than 90% [Figure2], while we only have 11 variables. But we can find several important correlations among these properties. For example, there is a negative correlation of -0.4698 between density and alcohol, indicating that higher density is associated with lower alcohol content.

Then, we are curious about the dependent variable which is a quality rating assigned by human evaluators. Notice that scores are subjective, euclidean distance is not fixed among scores. We treat quality as category variable and intend to do classification. After plotting a histogram based on quality, we figure out the dataset is imbalanced [Figure1]. The number of wines rated as 5 or 6 is significantly more than the rest. We split the dataset into bad (3 to 4) normal (5 to 6) good (7 to 8) and do a multi-class classification.

The data processing pipeline involves several steps to prepare the dataset for machine learning modeling. Firstly, the dataset is split into training (80%) and testing(20%) sets. After the initial split, the training set undergoes oversampling, specifically using the Synthetic Minority Over-sampling Technique (SMOTE). The oversampling process is followed by undersampling using the RandomUnderSampler to further balance the class distribution. Finally, a hybrid sampling approach is employed. Additionally, standardization is applied to both the training and testing sets to ensure that all features are on a similar scale.

The goal of our project is to compare performance of different models in certain metrics and to stand in the position of red wine manufacturer to conduct profit maximization. We aim to give suggestions to predict the quality of wine based on the 11 physicochemical features. Moreover, we intend to plot precision-recall curves for each model, specifically for 'good' and 'bad' categories. Based on the optimal curve and the provided profit for 'good' wine, we can perform profit maximization. In section 3, we will evaluate metrics and select hyperparameters. Then we will evaluate the performance of different models and determine the best one. Finally, we will conclude what we have done in this project and mention future work.

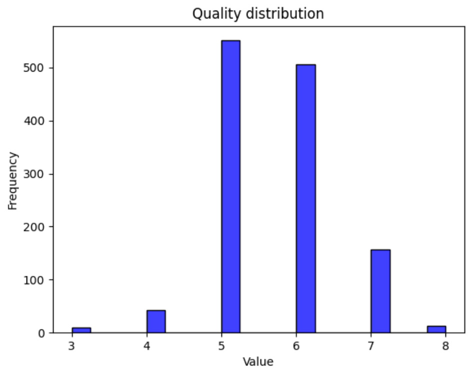


Figure 1: Quality distribution

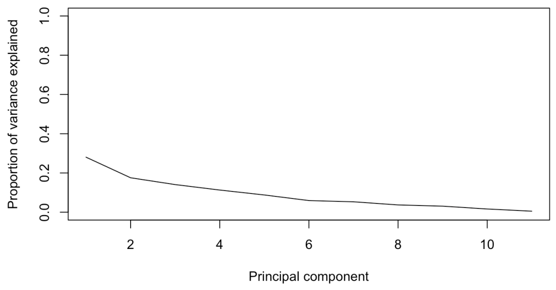


Figure 2: T.V.E

2 Error Metrics & Hyper-parameter Selection

Following the previously stated objectives, we consider two assumptions:

1) Due to limited data, the precision-recall trade-off is evident. We must choose either recall or precision at the cost of the other.

2) Given the high unit price of good wine and the significant negative impact of bad wine, standing in the position of a wine manufacturer, we aim to maintain a good reputation in selling our wine. Thus, we need to maximize the precision in predicting good wine while minimizing the mislabeling of bad wines into other groups.

Building upon the above discussion, we introduce three error metrics in our project:

(1) Balanced Accuracy:

$$BA = (\frac{TP_{bad}}{T_{bad}} + \frac{TP_{normal}}{T_{normal}} + \frac{TP_{good}}{T_{good}})/3 = \overline{Recall\ Rate}$$

(2) Precision in Good:

$$Precision_{good} = \frac{TP_{good}}{P_{good}}$$

(3) Recall in Bad:

$$Recall_{bad} = \frac{TP_{bad}}{T_{bad}}$$

To achieve superior performance in these metrics, we explore three different methods. Firstly, adjusting the threshold of classification proves to be less effective. For example, attempting to increase the precision of good may lead to a higher probability threshold, resulting in a decrease in both the number of true 'good' and true 'normal' in the predicted 'good', leading to a minimal rise in good precision but a significant drop in good recall and, consequently, less profit. Hence, this method is not considered.

Secondly, we employ a self-defined scorer and multiple hyperparameter selection packages. To enhance metric 1, we experiment with different models, given that most Python solvers aim for higher accuracy, and our training data is balanced across all three resample methods. For metrics 2 and 3, we utilize the `make_scorer` package from `sklearn.metrics` to define a metric function based on the recall rate of bad and precision rate of good:

$$score = w * Precision_{good} + (1 - w) * Recall_{bad}$$

We then perform hyperparameter selection using three methods: `GridSearchCV`, `RandomizedSearchCV`, and `hyperopt`, employing the defined scorer.

Thirdly, we implement two binary classifications, using one classifier to label bad or not bad, and another classifier to label good or not good, based on the best-performing three-classifiers on each of the two tasks. Further details are provided at the end of the model section.

3 Models & Results

3.1 Logistic Regressions

Multinomial Logistic Regression is a statistical method for multi-class classification. Unlike binary logistic regression, it handles scenarios with more than two classes. This method uses the Softmax function to combine binary logistic regression models, accommodating multiple categories without decomposing the problem. Here, I introduce the L1 (Lasso) and L2 (Ridge) penalties in multinomial regression to better handle strong feature correlations. Logistic regressions show good performance in classifying bad wine, achieving a recall rate of nearly 100%, although precision for bad wine is suboptimal.

3.2 Tree Methods

For tree methods, I applied three popular ones: decision tree, random forest, and XGBoost. Decision trees construct hierarchical trees using recursive binary splitting and Gini index. Random forest comprises multiple decision trees trained independently, and XGBoost builds trees sequentially, correcting errors made by previous ones. Tree methods generally outperform logistic regression, although they are less effective in labeling bad wines.

3.3 Final Model: Two Binary Classification

During the model selection of 3-class classification, I observed minimal mislabeling between good and bad wines. Therefore, I split the test data into bad or not and good or not, conducting binary classification twice using the two best models in terms of bad recall rate and good precision rate. The final model combines Lasso multi-logistic regression and random forest, proving its value with the best performance across three metrics [Figure8].

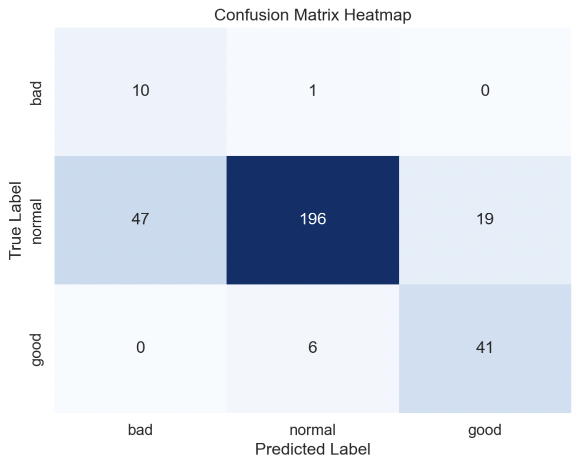
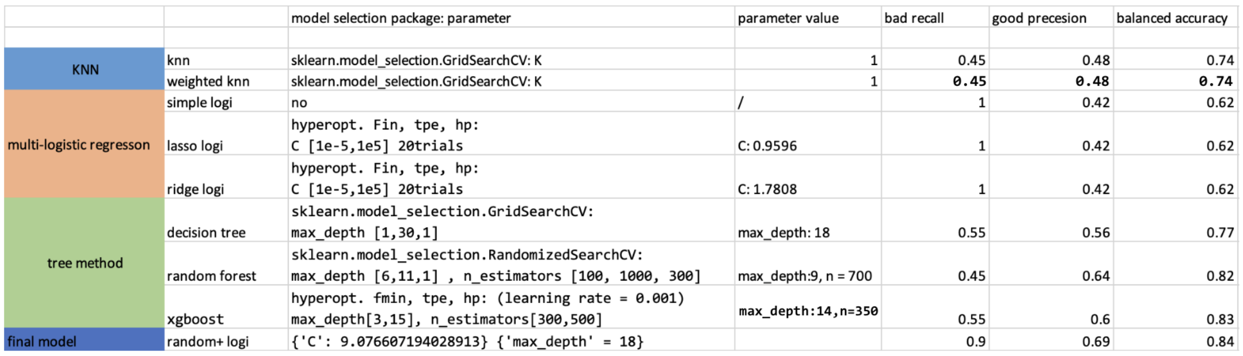


Figure 3: confusion matrix

3.4 Model Comparison

Following the error metric and cross validation techniques I mentioned above, I record the three error metrics of all my models [Figure9]. KNN operates by identifying the k-nearest data points to a given sample in the feature space, where proximity is often measured using Euclidean distance. KNN weighted assigns higher weight to points closer to the test point in the K nearest neighbor. However, these two methods did not perform well, leading to their exclusion in further discussion.



		model selection package: parameter	parameter value	bad recall	good precesion	balanced accuracy
KNN	knn	sklearn.model_selection.GridSearchCV: K	1	0.45	0.48	0.74
	weighted knn	sklearn.model_selection.GridSearchCV: K	1	0.45	0.48	0.74
multi-logistic regresson	simple logi	no	/	1	0.42	0.62
	lasso logi	hyperopt. Fin, tpe, hp: C [1e-5,1e5] 20trials	C: 0.9596	1	0.42	0.62
	ridge logi	hyperopt. Fin, tpe, hp: C [1e-5,1e5] 20trials	C: 1.7808	1	0.42	0.62
tree method	decision tree	sklearn.model_selection.GridSearchCV: max_depth [1,30,1]	max_depth: 18	0.55	0.56	0.77
	random forest	sklearn.model_selection.RandomizedSearchCV: max_depth [6,11,1] , n_estimators [100, 1000, 300]	max_depth:9, n = 700	0.45	0.64	0.82
	xgboost	hyperopt. fmin, tpe, hp: (learning rate = 0.001) max_depth[3,15], n_estimators[300,500]	max_depth:14,n=350	0.55	0.6	0.83
final model	random+ logi	{'C': 9.076607194028913} {'max_depth' = 18}		0.9	0.69	0.84

Figure 4: Model summary

3.4.1 Resample-wise

Across resample methods (Smote oversampling, random undersampling, hybrid resampling), oversampling performs best. The data quantity is insufficient for 3-class classification, making any resample method reducing the majority class detrimental to model fitting.

3.4.2 Model-wise

For 3-class models, there's a trade-off between high bad recall rate and good precision. Lasso, and random forest are top candidates, but none achieves a rate larger than 0.5 for all three metrics, motivating me to put forward the model with two binary classifications, which is considered as our final model.

4 Discussion and Conclusion

In this project, we explored various machine learning techniques to classify the quality of red wine, aiming to create a predictive model that could assist in automating quality control and enhancing the production process. We employed traditional algorithms like Logistic Regression, as well as more complex methods such as XGBoost. Our findings suggest that while each model has its strengths, there is a consistent challenge across all methods in accurately predicting the minority classes. This is indicative of the inherent difficulty of the task, compounded by the imbalanced nature of our dataset. Despite employing techniques like SMOTE to address this imbalance, the subtleties of the minority class remain elusive for our models to capture effectively.

It's important to recognize certain limitations in our approach. The complexity of wine quality assessment, which involves subjective human taste preferences, might not be fully captured by the features present in our dataset. Additionally, the oversampling technique, while balancing the class distribution, may have introduced synthetic noise that does not perfectly represent the true minority class instances.

Despite these challenges, the developed models can still be valuable in a real-world setting. They could serve as an initial screening tool to flag potential outliers in quality, thus streamlining the process before experts conduct a more nuanced assessment. This could lead to cost savings and efficiency improvements in the wine production industry.

To enhance the performance of our predictive models, we might have to consider multiple approaches. Acquiring a larger and more naturally balanced dataset stands as a primary objective, which may alleviate the current challenges posed by the imbalanced data and reduce the reliance on synthetic oversampling techniques. Concurrently, considering the ordinal nature of wine quality, the application of ordinal classification methods holds promise for potentially superior performance, as they can utilize the inherent order of quality ratings. Furthermore, we will explore hybrid models or more advanced neural network architectures that might capture the complexities of the dataset more effectively, possibly leading to breakthroughs in predictive accuracy and reliability.

5 Reference

UCI MACHINE LEARNING (2017) Red Wine Quality
https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009?rvi=1