# Classify quality of wine

Xinkai Wang, xinkaiwa@usc.edu
Dec 2, 2018

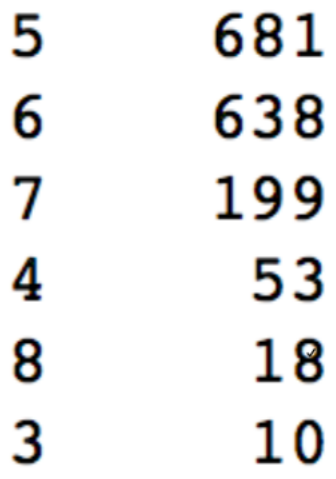
## 1. Abstract

Logistic regression, random forest and adaboost are three popular algorithms in supervised learning. Random forest generates decision trees by randomly selecting a limited number features from all available features for node splitting, and each tree cast a vote for the final decision. [1] By assigning weights to training samples, adaboost improves the performance of a weak classifier with rule incorrectly classified training sample will gain a larger weight. [1] These three algorithms are used to train multiclass classification models to classify quality of wine. I discuss performance of these algorithm and select the best model.

## 2. Introduction

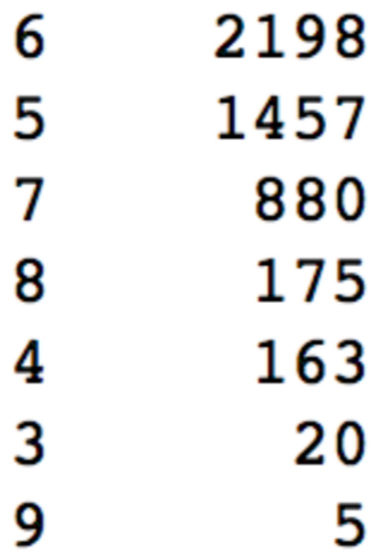
### 2.1. Problem Type, Statement and Goals

The project I finished was a multiclass classification problem to classify quality of wine. The data includes red wine and white wine, as figure 2.1.1 and figure 2.1.2 shows. Left column is label and right column is data scale, the larger label means its quality is greater. The goal is to model wine quality based on physicochemical tests. The problem is important because it is a multiclass badly imbalanced problem with the smallest class containing only 10 data and the largest class including 681 data. It has 11 features.



| | |
|---|---|
| 5 | 681 |
| 6 | 638 |
| 7 | 199 |
| 4 | 53 |
| 8 | 18 |
| 3 | 10 |

**Figure 2.1.1**: red wine quality classes



| | |
|---|---|
| 6 | 2198 |
| 5 | 1457 |
| 7 | 880 |
| 8 | 175 |
| 4 | 163 |
| 3 | 20 |
| 9 | 5 |

**Figure 2.1.2**: white wine quality classes

## 2.2. Literature Review

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.

## 2.3. Prior and Related Work – None

## 2.4. Overview of Approach

I used multinomial logistic regression, random forest and adaboost algorithm. Initially, I split data into train, validation and test groups with ratio of 6:2:2. Then recursive feature elimination (RFE) and extra tree classifier were used to select features. Next, for each method, cross validation was applied to training data to select parameters. What should be mentioned is that instead normal accuracy score, I used balanced accuracy score as evaluation rule. I used validation data to choose best selected features among the best 8, 9, 10 and all features, and compared outcome of RFE with that of extra tree classifier. Finally, test data was used to do model evaluation.

## 3. Implementation

## 3.1. Data Set

| Feature names | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | |
|---|---|---|---|---|---|---|
| type | float | float | float | float | float | |
| min | 4.60 | 0.12 | 0.00 | 0.90 | 0.01 | |
| mean | 8.32 | 0.53 | 0.27 | 2.54 | 0.09 | |
| max | 15.90 | 1.58 | 1.00 | 15.50 | 0.61 | |
| count | 1599 | 1599 | 1599 | 1599 | 1599 | |
| Feature names | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol |
| type | float | float | float | float | float | float |
| min | 1.00 | 6.00 | 0.990 | 2.74 | 0.33 | 8.40 |
| mean | 15.87 | 46.47 | 0.996 | 3.31 | 0.66 | 10.42 |
| max | 72.00 | 289.00 | 1.004 | 4.01 | 2.00 | 14.90 |
| count | 1599 | 1599 | 1599 | 1599 | 1599 | 1599 |

**Table 3.1.1**: data describe of red wine

| Feature names | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | |
|---|---|---|---|---|---|---|
| type | float | float | float | float | float | |
| min | 3.80 | 0.08 | 0.00 | 0.60 | 0.01 | |
| mean | 6.85 | 0.28 | 0.33 | 6.39 | 0.05 | |
| max | 14.20 | 1.10 | 1.66 | 65.80 | 0.35 | |
| count | 4898 | 4898 | 4898 | 4898 | 4898 | |
| Feature names | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol |
| type | float | float | float | float | float | float |
| min | 2.00 | 9.00 | 0.987 | 2.72 | 0.22 | 8.00 |
| mean | 35.31 | 138.36 | 0.994 | 3.19 | 0.49 | 10.51 |
| max | 289.00 | 440.00 | 1.039 | 3.82 | 1.00 | 14.20 |
| count | 4898 | 4898 | 4898 | 4898 | 4898 | 4898 |

**Table 3.1.2**: data describe of white wine

## 3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

For red wine data, I split data into training data, validation data and test data with ratio 6:2:2 and each group separated into features X and label y. I got r_X_train, r_y_train, r_X_val, r_y_val, r_X_test, r_y_test. StandardScaler library was used to standardize features. I obtained scal_X_train, scal_X_val, scal_X_test. Because this data is badly imbalanced, I also used library SMOTE to resample my data. Since the largest class ('quality'=5) has 681 data while the smallest class ('quality'=3) has only 10 data, after splitting data into training, validation and test, for training data, I used One Side Selection to under-sample majority classes, then used SMOTE to oversample other classes. X_res, y_res were obtained as resampled X_train and y_train.

| | | | | | | |
|---|---|---|---|---|---|---|
| Normal data | r_X_train | r_y_train | r_X_val | r_y_val | r_X_test | r_y_test |
| Standardized data | scal_X_train | r_y_train | scal_X_val | r_y_val | scal_X_test | r_y_val |
| Resampled data | X_res | y_res | r_X_val | r_y_val | r_X_test | r_y_test |

. **Table 3.2.1**: red wine data

## 3.3. Dataset Methodology

Library RFE with LogisticRegression in multinomial form and ExtraTreeClassifier were applied to select most important features in scale of 8, 9, 10 and remaining all features.

Next step was cross validation. For multinomial logistic regression model, I used library GridSearchCV (all cross validation used 5-fold validation) to select parameter for model with training data, for example, in case of normal data, r_X_train and r_y_train is training data. Selected parameter was C (=1/lambda) within range(0.001, 0.01, 0.1, 0, 1, 10, 100, 1000). The above process was implemented in a loop to choose selected 8, 9, 10 features and all features which is the best data to use in training. The model was used to predict validation data e.g. r_X_val whose outcome as y_pred. Applying library balanced_accuracy_score to evaluate r_y_val and y_pred, parameters getting the highest score were the best parameters.

Similarly in process of random forest and adaboost. However, because library RandomForestClassifier has many parameters, I first used library

RandomizedSearchCV to approximate ranges of parameters, then GridSearchCV was used to select the final parameters. To save time, I annotated the RandomizedSearchCV part in my code.
I used the best parameters selected by random forest as the parameter for DecisionTreeClassifier in adaboost.
The test set was only used in the final evaluation of models (out of sample performance).
All evaluation of selection was based on balanced_accuracy_score.

## 3.4. Training Process

(1) Logistic Regression

- $f(k,i) = \underline{W}^T\underline{X} = w_{0,k} + w_{1,k}x_{1,i} + \ldots + w_{M,k}\, x_{M,i}$ (maximum of M is 11 in this project) X is data vector, W is weight vector.
  It is a model used to predict probabilities of outcomes of categorically distributed dependent variables.

- Logistic Regression in multinomial form is simple method for baseline.

- In python library Logistic Regression, set parameter “multi_class” = “multinomial”, “solver” = “newton-cg”, “class_weight” = “balanced”. Newton-cg is newton conjugate gradient. Balanced “class_weight” penalize mistakes on the minority class. Parameter “C” was chosen by cross validation.

- I used cross validation to select parameters and feature selection in validation process to avoid overfitting, since the lower feature dimension was selected, the lower probability that the model was overfitting.

(2) Random Forest

- Random forest algorithm is based on CART (decision tree). Random forest is to combine many decision trees in one model. Each decision tree in the forest considers a random subset of features when forming questions and only has access to a random set of the training data points. [2] This increases diversity in the forest resulting in more robust overall predictions. [2] When makes prediction, for classification problem, those decision trees make a vote to get the class label.

- The hierarchical structure in random forest works well in imbalanced problem by allowing to learn signals from all classes.

- Parameters I set in selection contained "n_estimator" (par1) , "max_depth" (par2), "min_sample_split" (par3), "min_sample_leaf" (par4), "max_features" (par5), class_weight" (par6). In first round, use RandomizedSearchCV (n_iter=50, cv=5, scoring=balanced_accuracy_score) with par1 range(200,1100,100), par2 range(10,70,10), par3 in list [2,5,10], par4 in list [1,2,4], par5 in list ['auto', 'sqrt'], par6 equaled to 'balanced'. This took long time.
  Selected best parameters were par1=500, par2=30, par3=5, par4=4, par5='sqrt', par6='balanced'. Then I use GridSearchCV (cv=5, scoring=balanced_accuracy_score) with par1 in list [400, 500, 600], par2 in list [20, 30, 40], par3=5, par4=4, par5 in list ['auto', 'sqrt'], par6='balanced'.

- I used cross validation on parameter "max_depth" and feature selection to avoid overfitting. Validation set was split out of data before the use of cross validation.

(3) Adaboost

- Adaboost creates a highly accurate prediction rule by combining many relateively weak rules. [3] It was the first practical boosting algorithm and remains one of most widely used model in numerous fields. [3]

- Adaboost algorithm would yield the best performance in my experiences.

- I selected DicisionTreeClassifier as the base classification method in adaboost. What tricks was that I used the best parameters outcome from above random forest as the parameters for the decision tree. Parameters in my selection were "base_estimator" (par1), "n_estimators" (par2), "learning_rate" (par3), "algorithm" (par4). GridSearchCV (cv=5, scoring=balanced_accuracy_score) was used with par1 in list [None, base decision tree, selected parameters decision tree], par2 in list [25,50,75], par3 in list [0.01,0.05, 0.1] and par4 in list ['SAMME', 'SAMME.R'].

- I used feature selection and cross validation on parameter selection to avoid overfitting.

## 3.5. Model Selection and Comparison of Results

I combined training balanced accuracy and test balanced accuracy as main score with also considering other scores in table to select best model.
From table 3.5.1, 3.5.2 and 3.5.3, random forest performs best.
From table 3.5.4, 3.5.5 and 3.5.6, removing minority classes quality=3&8, adaboost performs best.
Table 3.5.7 is xgboost performance on data removing minority classes quality=3&8. Adaboost works better.
Then use white wine data after deleting minority classes quality=3&9, from table 3.5.8 and 3.5.9, adaboost performs better than random forest.
In conclusion, adaboost with decision tree as base classifier performs best in these tree models.

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| Normal | 9 | 0.538 | 0.407 | 0.362 | 0.449 | 0.406 |
| After standardizing | 11 | 0.559 | 0.399 | 0.326 | 0.425 | 0.375 |
| After resampling | 10 | 0.631 | 0.631 | 0.374 | 0.487 | 0.456 |

Table 3.5.1: Logistic regression outcome

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| normal | 11 | 0.969 | 0.925 | 0.350 | 0.619 | 0.625 |
| After standardizing | 8 | 0.955 | 0.897 | 0.354 | 0.591 | 0.597 |
| After resampling | 10 | 0.952 | 0.913 | 0.382 | 0.564 | 0.550 |

Table 3.5.2: random forest outcome

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| normal | 11 | 0.98 | 0.96 | 0.316 | 0.639 | 0.656 |
| After standardizing | 8 | 0.98 | 0.96 | 0.340 | 0.653 | 0.666 |
| After resampling | 8 | 0.97 | 0.94 | 0.302 | 0.609 | 0.622 |

Table 3.5.3: adaboost

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| normal | 11 | 0.522 | 0.466 | 0.570 | 0.486 | 0.460 |
| After standardizing | 10 | 0.536 | 0.482 | 0.569 | 0.520 | 0.502 |
| After resampling | 10 | 0.562 | 0.562 | 0.605 | 0.523 | 0.502 |

Table 3.5.4: logistic regression after data adjust (remove classes quality=3,8)

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| normal | 11 | 0.957 | 0.930 | 0.470 | 0.619 | 0.632 |
| After standardizing | 9 | 0.950 | 0.925 | 0.459 | 0.606 | 0.619 |
| After resampling | 9 | 0.963 | 0.963 | 0.510 | 0.586 | 0.584 |

Table 3.5.5: random forest after data adjust (remove classes quality=3,8)

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| normal | 10 | 0.98 | 0.97 | 0.480 | 0.638 | 0.652 |
| After standardizing | 10 | 0.98 | 0.97 | 0.345 | 0.554 | 0.584 |
| After resampling | 10 | 0.98 | 0.97 | 0.480 | 0.638 | 0.652 |

Table 3.5.6: adaboost after data adjust (remove classes quality=3,8)

I used popular xgboost to compare with adaboost. Adaboost performs better than xgboost.

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| Normal | 11 | 1.0 | 1.0 | 0.476 | 0.644 | 0.657 |
| After standardizing | 11 | 1.0 | 1.0 | 0.437 | 0.607 | 0.625 |
| After resampling | 11 | 1.0 | 1.0 | 0.482 | 0.598 | 0.594 |

Table 3.5.7: xgboost after data adjust (remove classes quality=3,8)

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| Normal | 11 | 0.964 | 0.934 | 0.572 | 0.644 | 0.645 |
| After standardizing | 10 | 0.962 | 0.930 | 0.537 | 0.624 | 0.626 |
| After resampling | 10 | 0.978 | 0.978 | 0.536 | 0.580 | 0.570 |

Table 3.5.8: white wine random forest after data adjust (remove classes quality=3,9)

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| Normal | 11 | 0.98 | 0.96 | 0.539 | 0.657 | 0.663 |
| After standardizing | 11 | 0.97 | 0.95 | 0.520 | 0.648 | 0.655 |
| After resampling | 9 | 0.97 | 0.94 | 0.528 | 0.647 | 0.652 |

Table 3.5.9: white wine adaboost after data adjust (remove classes quality=3,9)

This multiclass non-linear separable data is hard to show decision boundary in 2D or 3D.

## 4. Final Results and Interpretation

Selected model is adaboost with parameters

“base_estimator”=DecisionTreeClassifier(class_weight=’balanced’, criterion=’gini’, max_depth=10, max_feature=’sqrt’, min_sample_leaf=8, min_sample_split=10) ;

”n_estimators”=75 ;

“learning_rate”=0.1 ;

“algorithm”= ‘SAMME’ .

Final performance is shown in table 3.5.3, table 3.5.6 and table 3.5.8. An estimate of out of sample performance includes test balanced accuracy, test f1 score and test accuracy.

## 5. Summary and conclusions

For some badly imbalanced problem, especially without large amount of data, it is hard to obtain a high-quality model. In practice, ignoring those minority classes improves model significantly or merging minority class with majority class to make data more balanced. One useful method to deal with imbalanced data is that first use One Side Selection (OSS) to under-sample majority classes, then apply Synthetic Minority Over-Sampling Technique (SMOTE) to oversample minority classes. From tables of 3.5, resampled data outperforms than standardized data slightly. What would be interesting to do next is to find better model to deal with badly imbalanced multiclass problem without a large number of data. I assume deep learning can train a better model.

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| Normal | 9 | 0.389 | 0.449 | 0.362 | 0.449 | 0.406 |
| After standardizing | 11 | 0.375 | 0.402 | 0.326 | 0.425 | 0.375 |
| After resampling | 11 | 0.419 | 0.488 | 0.380 | 0.444 | 0.406 |

Table 3.5.1: Logistic regression outcome

| Data type | Selected features number | Training balanced accuracy | Training accuracy | Test balanced accuracy | Test f1 score | Test accuracy |
|---|---|---|---|---|---|---|
| normal | 11 | 0.536 | 0.464 | 0.570 | 0.486 | 0.460 |
| After standardizing | 10 | 0.499 | 0.480 | 0.569 | 0.520 | 0.502 |
| After resampling | 10 | 0.532 | 0.468 | 0.605 | 0.523 | 0.502 |

Table 3.5.4: logistic regression after data adjust (remove classes quality=3,8)

## 6. References

[1] X. Miao and J. S. Heaton, "A comparison of random forest and adaboost tree in ecosystem classification in east mojave desert," in *2010 18th International Conference on Geoinformatics,* 2010, . DOI: 10.1109/GEOINFORMATICS.2010.5567504.

[2] “random forest simple explaination” by William Koehrsen, Dec 27, 2017
Available: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

[3] Schapire R.E. (2013) Explaining AdaBoost. In: Schölkopf B., Luo Z., Vovk V. (eds) Empirical Inference. Springer, Berlin, Heidelberg