# 

# 

# 

Project 3: Wine Quality Classification

BUS 212: Analyzing Big Data II

Group Member: Xin Chen, Yutong Li

05/05/2021

## **I. Problem Statement**

In this project, we will perform classification on red wine quality. The red wine dataset was developed by P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.[[1]](#footnote-0)

In this dataset, we have input variables based on physicochemical tests, such as fixed acidity, volatile acidity, etc. Also, there is an output variable based on sensory data, called quality (score between 0 and 10). We will first transform the output variable into a binary variable, then apply classification on the dataset to see if we can accurately classify the red wine quality.

Data Source:

Input variables (based on physicochemical tests):

1 - fixed acidity

2 - volatile acidity

3 - citric acid

4 - residual sugar

5 - chlorides

6 - free sulfur dioxide

7 - total sulfur dioxide

8 - density

9 - pH

10 - sulphates

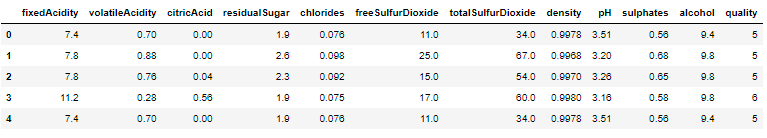
11 - alcohol

Output variable (based on sensory data):

12 - quality (score between 0 and 10)

## **II. Data Preparation**

We use head() function to check if we have successfully loaded the data.



Data Cleaning Pipeline:

This pipeline is quoted from <https://towardsdatascience.com/cleaner-data-analysis-with-pandas-using-pipes-4d73770fbf3c>. This pipeline includes functions that could handle missing data, remove outliers, and change categorical data types. We modified and edited this pipeline according to what we need for this data.

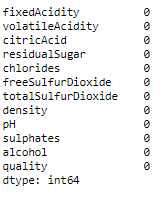
#### **2.1 Verify Data**

We use info() for checking more detailed information about the data.

#### 

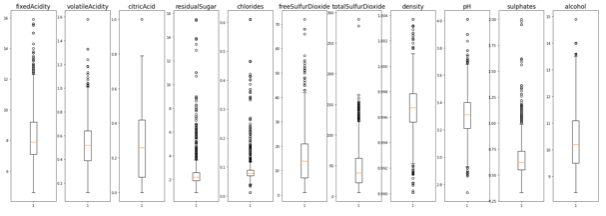
#### **2.2 Missing Data**

Using isnull().sum(), we check that we do not have missing value.



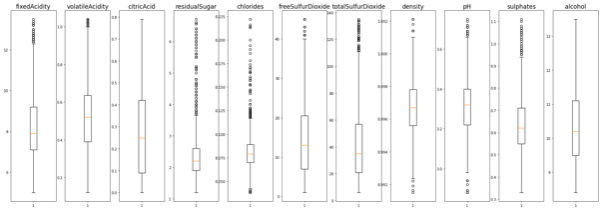
#### **2.3 Inspect Outliers**

Next, we used boxplots to see if there are outliers in some numerical field that would potentially hurt our following prediction. The graph is shown below:

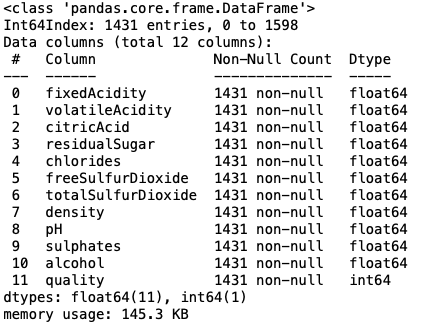


Notice that many data columns have outliers. There are many commonly-used ways of handling outliers. For example, we could drop values below 0.05 quantile or above 0.95 quantile, or we could keep data only between Q1-1.5 IQR and Q3+1.5 IQR. But here, we decided to include data within only three standard deviations to ensure that we have enough data for further analysis.

Here, we used the pipeline we just mentioned, and let’s look at the boxplots again and see how well our pipeline works:



The graphs look much better now. Let’s use the info() function again to check our dataset now after cleaning.

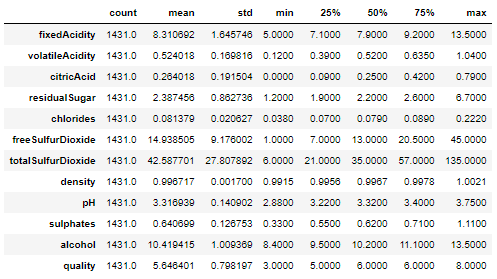


#### **2.4 Set up Binary Classifier**

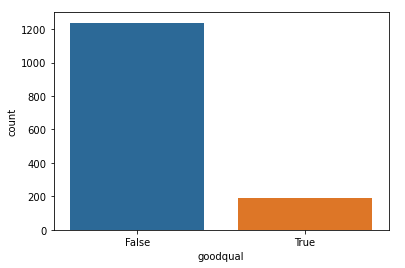
We define quality > 6.5 as good quality, and quality <= 6.5 as poor quality. Good quality = 1 and poor quality = 0.

## **III. Exploratory Data Analysis**

To show the detailed distribution of each feature in the dataset, we use describe().transpose()

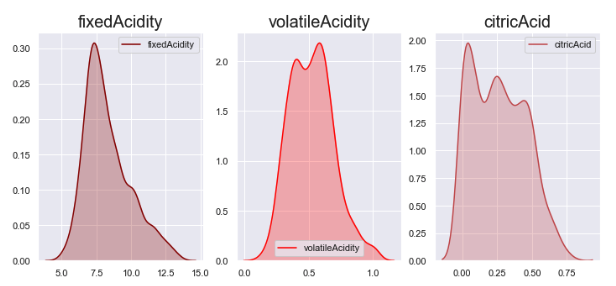


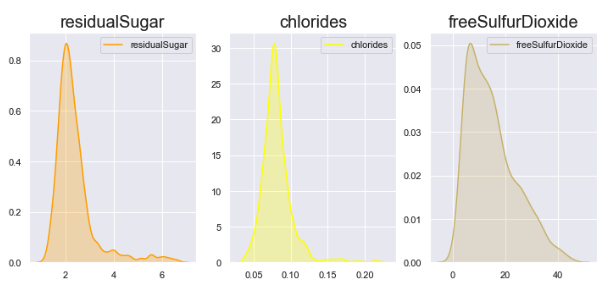
Barplot of goodqual:

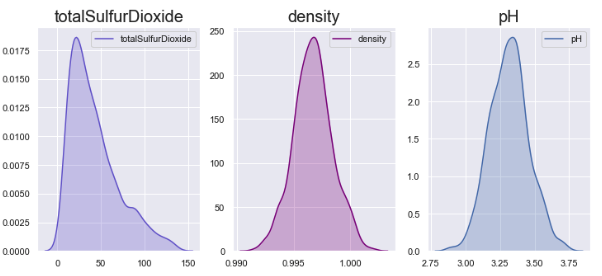


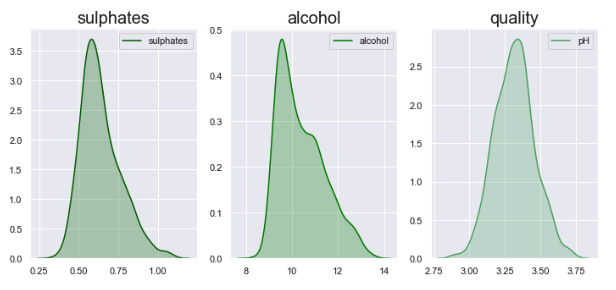
Notice that here we did not have a balanced target variable. Usually we could use some bootstrap sampling to get balanced data. But here, for this project, we wanted to use all data possible since we do not have a very large dataset.

Histogram of Categorical Variables:



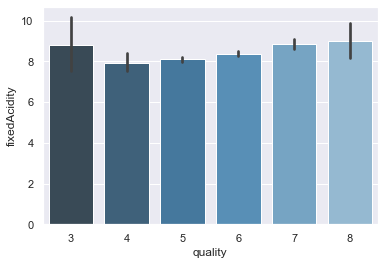
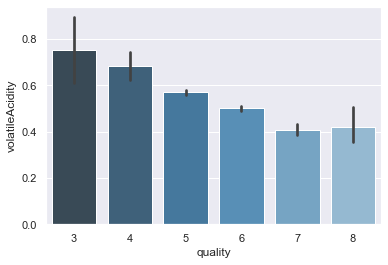


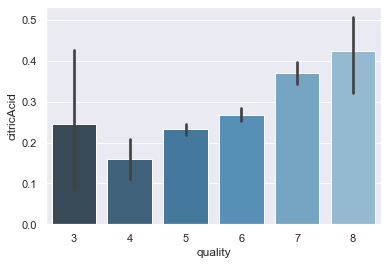
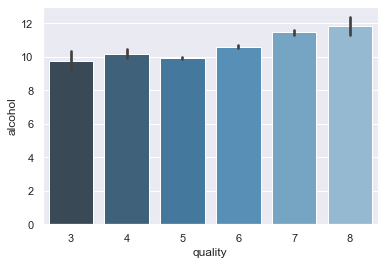
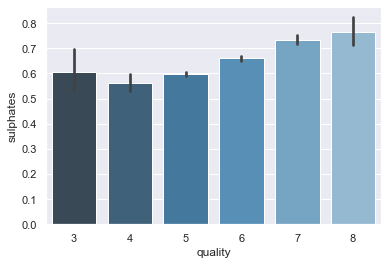
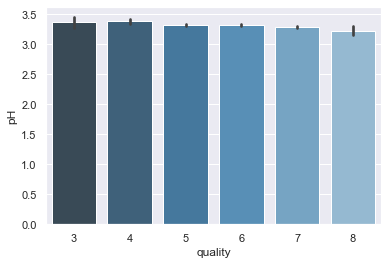
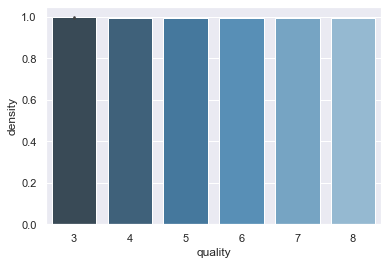
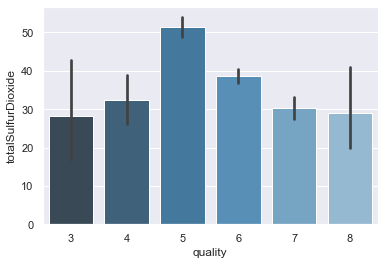
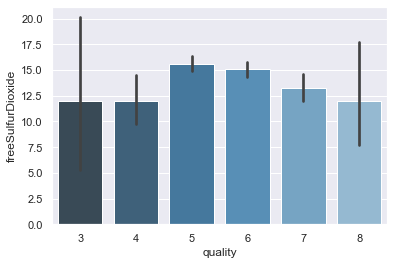
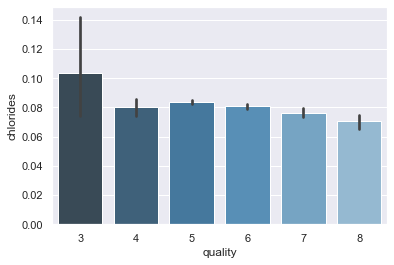
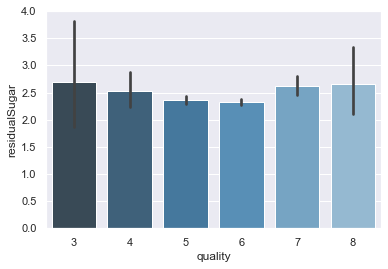




Pairwise Relationship:

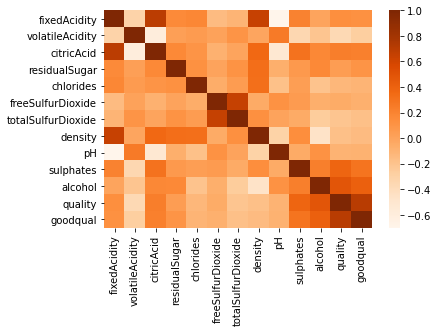
Here, we first used a bar chart to show the pairwise relationship between each qualitative variable and the goodqual variable. We chose bar chart over scatter plot is due to the fact that goodqual is a categorical variable.

****

****

From the pairwise relationship plots above, we see that 'fixedAcidity', 'residualSugar', 'density', and 'pH’ don’t show visible influences on different quality levels, so we decide to drop these features in the later data processing section.

Heatmap for correlations between features



## **IV. Classification & Performance Evaluation**

Now with data ready and explored, we can move onto our next stage - use machine learning models to make classifications. Here, we are going to try several models, and based on the performance measurement, we will choose the best model.

About Standardization:

Most models we have chosen do not require standardization, such as Linear Regression, Decision Trees, and Random Forests. However, we will need to standardize for the KNN model, as standardization matters when calculating Euclidean distance. Therefore, we chose normalization, which also allows us to move the points centered at zero. Here, we standardize the whole dataset, while in practice, scaling only using the training set will be better.

About Principal Components Analysis (PCA):

Before jumping into our regression, we also wanted to first specify our decision on not using PCA for these models. From the heatmap in our earlier section, we have shown that the variables are not strongly correlated, hence PCA will not be really effective in reducing the dimension of data. Also, we would like to look at each variable’s effect on goodqual separately, so we don't want to transform them to new components, which would be hard to interpret.

Performance Evaluation Metrics:

Here, as we mentioned, we will use the score() function (R^2) to measure both the training and testing accuracy.

Cross Validation:

For many models, we will use a cross validation method to ensure stability of our model. And we will mostly use a shufflesplit for 25 times ith 0.2 testing data and 0.8 training data. For some specific model, due to its computational complexity, we will only use a one-time train/test split. And that’s why we first split our dataset into training data and testing data for further usage.

Hyperparameter Tuning:

For hyperparameter tuning, our choice is by splitting the training data into a “real” training set and a validation set, and measuring the performance on the validation set in order to choose the best hyperparameter.

Naive Benchmark:

We have 0.865 as our Naive Benchmark.



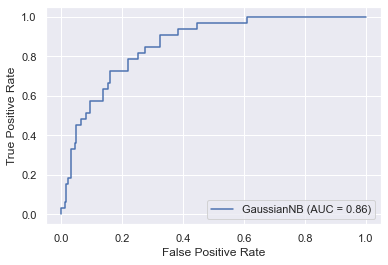
#### **4.1 Naive Bayes**

Gaussian Naive Bayes is one of the simple models we could first try which is based on a probabilistic algorithm. Here we have result:

##### Accuracy



##### ROC Curve



##### Cross Validation Machinery:



#### **4.2 Linear Discriminant**

##### Cross Validation Accuracy:

****

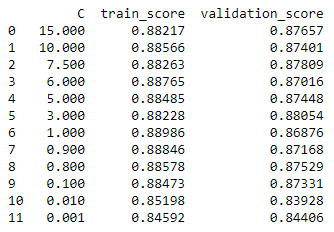
#### **4.3 Logistic**

##### Without Penalty:



##### With Penalty:

###### Grid Search for the best C value using validation data:

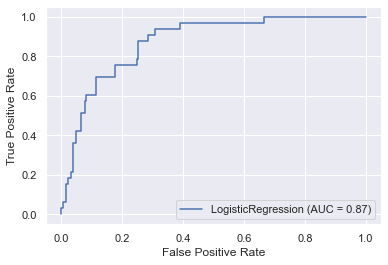


The best model is(are) the one(s) below:



And we get precision = 0.63, recall = 0.36

###### ROC Curve



##### Comparison:

0.87581 > 0.86977. From the above, we could see that the Logistic model without penalty actually performs better than the one with penalty.

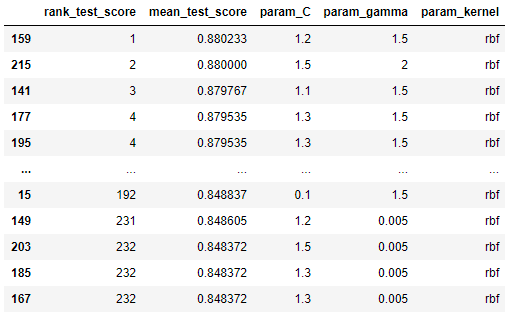
#### **4.4 SDG Classifier**

##### Cross Validation Accuracy:



#### **4.5 SVM**

##### Grid Search for the Best Parameters:

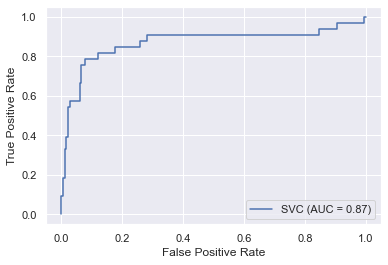


And the best parameters are C=1.2 and gamma=1.5.

##### Cross Validation Accuracy:

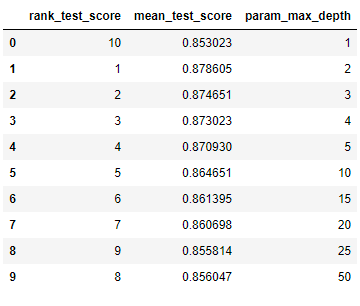


##### ROC Curve



#### **4.5 Decision Tree**

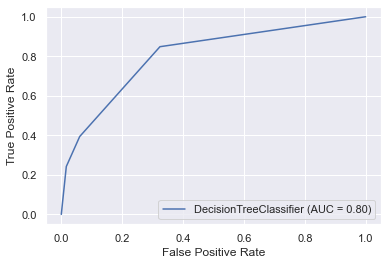
##### Grid Search for the Best Parameters:



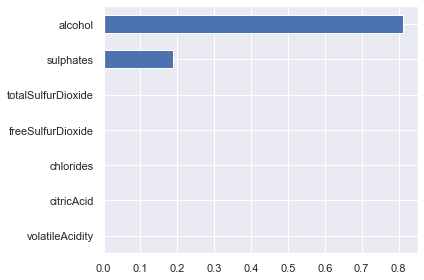
Using the best parameter max\_depth=2, we compute the mean accuracy:



##### ROC Curve



##### Plot Feature Importance



## **V. Ensemble Learning Algorithms**

Now, we could use several ensemble learning algorithms to improve our model performances. Ensemble models are more robust and versatile than individual models in the sense that they aggregate many models together and yield better results.

#### **5.1 Bagging (Bootstrap Sampling)**

We will train an ensemble of 1000 Decision Tree Classifiers, and each is trained on 429 (half training) instances randomly sampled with replacement in the training set. This could create a relatively larger bias, but lower variance, and thereby improve the performance.

##### Mean Accuracy



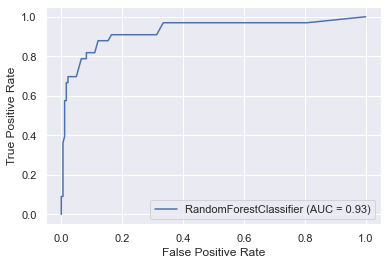
#### **5.2 Random Forest (Bagging)**

After grid search for the best parameters, we get the best parameters: max\_depth=15, max\_features=2, and n\_estimators=100. What makes Random Forest different from our 5.1 bootstrap sampling model is that it also limits the features at each node to be selected only from a smaller subset to enable its model diversity.

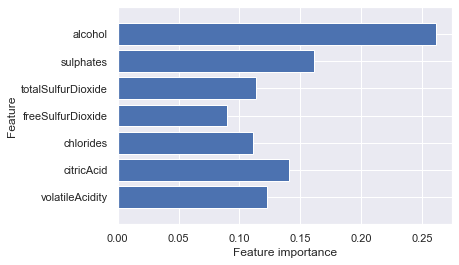
##### Mean Accuracy



##### ROC Curve

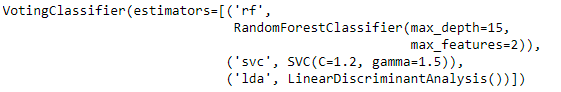


##### Plot feature importances

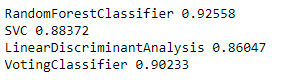


The graph also visualizes what features contribute more, but features look more diversified than trees.

#### **5.3 Voting Classifier**

Now we could use a voting classifier. This is one of the ensemble methods that combines several diversified models together, and we aim to produce a strong learner. This indeed tends to generate a great performance if all classifiers are independent, making uncorrelated errors. This is based on the law of large number theorem or the "wisdom" of the crowd. Here, we choose some models we got from the earlier section -- the best Random Forest Classifier, the best SVC, and Linear Discriminant model.

##### Accuracy for separate models and the voting classifier



We see that this voting classifier is improving on the top of LDA and SVC, however, RandomForestClassifier as an ensemble method itself still performs a little better than the voting classifier.

#### **5.4 Gradient Boost**

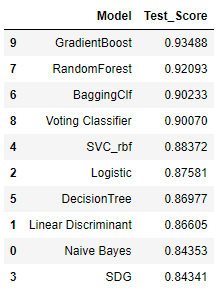
We set max\_depth = 3 and n\_estimators = 500.

##### Mean Accuracy



## **VI. Model Comparison**

Let's create a Pandas dataframe to display all scores that we have stored, then sort them descendingly based on the testing set performance.

****

From the above sorted dataframe, we could see that **GradientBoost** is the so far the best model which generates a testing accuracy of 0.93488. Pretty good score! We could also notice that Ensemble Learning methods such as **RandomForest** and Bagging Classifier are also performing pretty well. On the other hand, the **SDG** classifier has the worst performance.

## **VII. Unseen Data Classification**

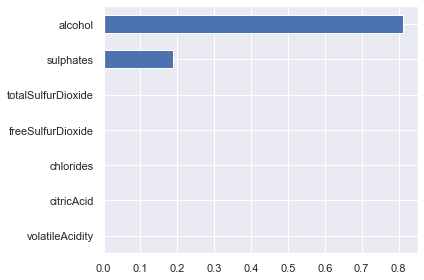
From the above performance comparison, we chose the top three models, and train the remaining unseen data. The performance is shown below:

|  | Mean training accuracy | Mean testing accuracy |
| --- | --- | --- |
| Gradient Boost | 1.0 | 0.86872 |
| Random Forest | 1.0 | 0.90223 |
| Bagging | 0.96365 | 0.89385 |

Based on the unseen data classification, Random Forest classifier outperforms other classifiers with a 0.90223 mean testing accuracy.

## **VIII. Conclusion**

As we mentioned before, Ensemble Learning models seem to generate the best results in general compared to individual models. Also if we look at the feature importance plot in section 4.5:



It actually shows us that only alcohol and sulphate features are of great importance in determining the Red Wine quality, while other features might be related but not that crucial.

## **IX. Reflection**

For future study, one thing we could modify is to shuffle and resample the data as we have less goodqual RedWine in the dataset. Also, we could also consider Neural Network as one way to train our data.

1. 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. [↑](#footnote-ref-0)