



CS 521 Information Structures With Python

Spring 2022

Wine Quality Analysis Report

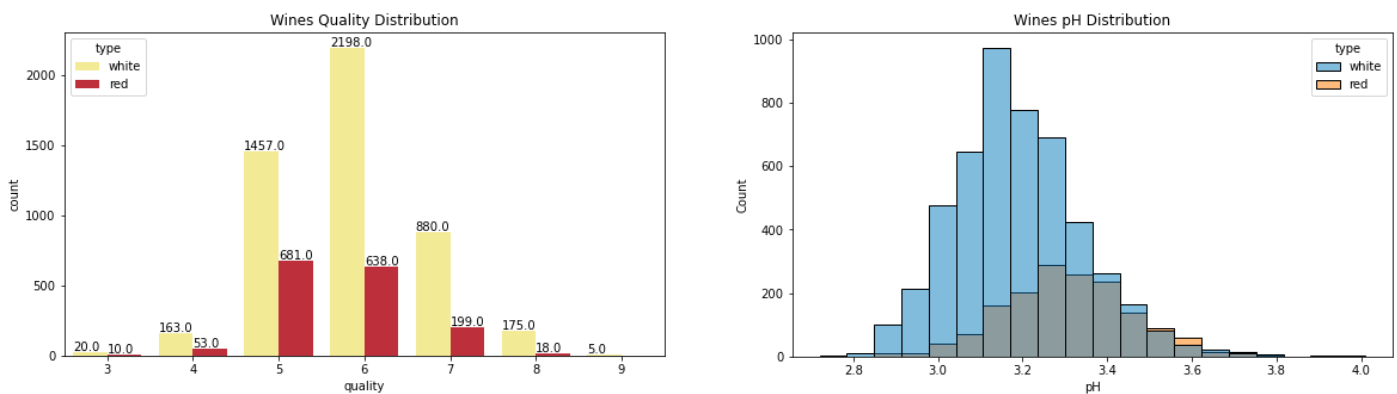
Xiangxian Song, Weiyao Xu, Longyuan Tang

May 4th, 2022

Analysis Process

(Methods and Techniques you used along with "brief" explanations of how they should work)

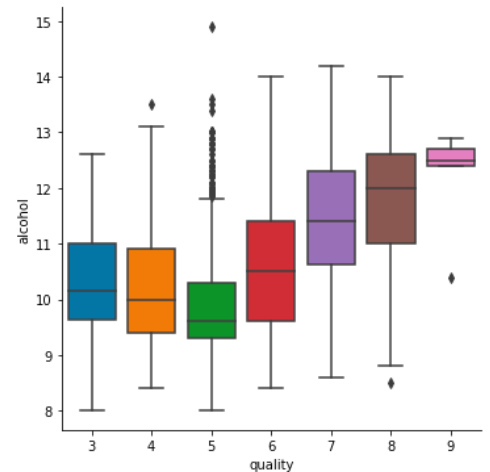
First of all, we need to have a general understanding of the data. After knowing the attribute information of the original data, we combined the red wine and white wine and did some visualization. We use matplotlib.pyplot and seaborn python libraries to create plots of different dimensions, such as heatmaps, boxplots, scatterplots, etc. Through the graphs, we got a preliminary understanding of some basic characteristics of wine, the commonalities and differences between two wines, as well as the distribution of different indicators and their relationship with quality.



The top left plot shows the quality distribution of red and white wine, we can clearly see that the quality of wine is mainly concentrated in 5 and 6. Among them, the quality range of White wine is more extensive, ranging from 3 to 9, mainly concentrated in 6, and the total number of wine is 1.5 times that of quality 5; Red wine's distribution of quality 5 and quality 6 is relatively similar, but significantly more than other quality rate. And through the top right plot which shows the pH distribution of red wine and white wine, we can find that the pH of the two wines is similar, and the white wine is generally more acidic. Most white wines have a pH in the 3.0-3.2 range, while most red wines have a pH in the 3.2-3.4 range. Also, in the range of 3.8-4.0 is almost red wine.

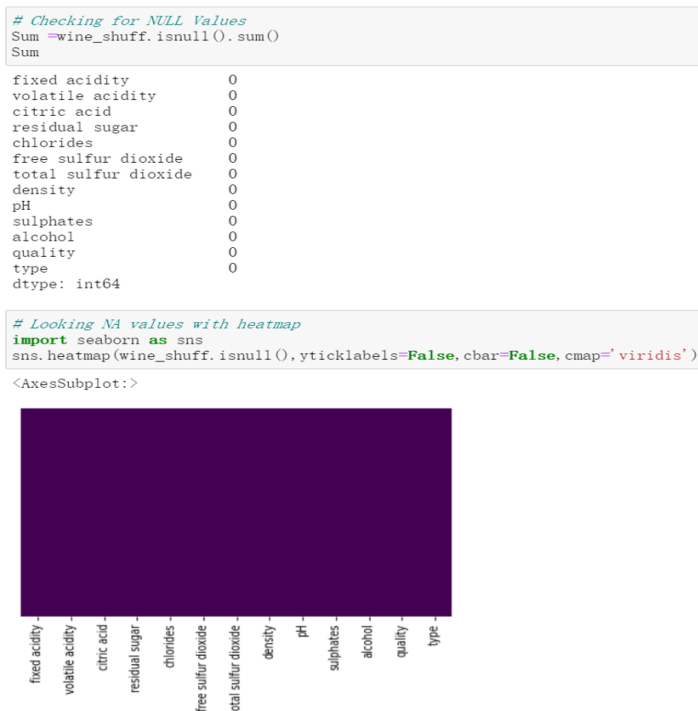
To go further, after general checking the potential relationships between different data attributes through heatmap. we also explored the relationship between individual index and quality. For example, the boxplot on the right shows the relationship between wine quality and alcohol content. we

can notice that the average alcohol content of quality 3-5 decreases, while the average alcohol content of quality 5-9 gradually increases. In the part of quality 5 and up, the higher the quality, the higher the alcohol content. Among them, the alcohol content of wine with quality 5 has a wider distribution range, and the alcohol content of wine with quality 9 is more concentrated than other qualities.



In the next part, we performed regression analysis.

Regression



Before modeling, we first need to check whether the data contain NA values. If there are a lot of NA values in the data, it means that we will lose much helpful information in the process of data mining modeling. Also, data containing NA values can confuse the modeling process, resulting in unreliable output. Such uncertainties will also be more significant, and the law implied in the model will

be more challenging to grasp. So, we first use `isnull().sum()` function and heatmap to detect NA values. Fortunately, our data is very clean, with no missing values.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=521)
print('train/test shapes:')
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)

train/test shapes:
(5197, 11) (1300, 11) (5197, 1) (1300, 1)
```

Before using machine learning algorithms, usually, we need to divide the dataset into a training set and a test set. When assigning training and test sets, the smaller the data in the test set, the less accurate the estimation of the model's generalization error will be. Therefore, we need to make trade-offs when dividing the dataset. Moreover, we should ensure that the divided data should be random; otherwise, the model will be useless. So based on the size of the entire dataset, we set the division ratio of training set data and test set data to 8:2. The method we use is the `train_test_split` method provided by sklearn. We use 80% of the data as the training set to fit the parameters in the model and then put the trained model on the remaining 20% of the data for testing to obtain the performance indicators of the model.

Linear Regression

```
# fit the linear model and test the score
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
mrl=linear_model.LinearRegression()
mrl.fit(x_train, y_train) # the training set is used to build the model
```

LinearRegression()

```
predl=mrl.predict(x_test) # the test set is used to predict
predl
```

```
array([[5.04367843],
       [5.37832657],
       [5.83554923],
       ...,
       [5.56984899],
       [5.19750126],
       [5.42343821]])
```

```
# compare the predicted value with the true value, and rate them
test_score = mrl.score(x_test, y_test)
print('test score of model = ', test_score) # R_squared: the regression line explains 29% of the total variation in the response values.
```

```
test score of model = 0.2945923833563917
```

```
# MSE: Mean Squared Error
from sklearn.metrics import mean_squared_error, r2_score, precision_score
print("The MSE of mrl is: ", mean_squared_error(y_test, predl))
```

```
The MSE of mrl is: 0.5245373471275777
```

To further explore the relationship between wine quality and other variables, we will perform a Multiple Regression Analysis. Set wine quality as the dependent variable and other variables as independent variables. To achieve our goal, we use the `linear_model.LinearRegression()` function to build a regression model. Then we use the test set to test the model to get the performance indicators of the model. We choose R^2 and MSE to judge the quality of the model. R^2 is called the coefficient of determination. It is used to determine the degree of fit. Its value range is $[0,1]$. If the result is 0, the model fits poorly; if the result is 1, the model is error-free. The larger the R-Squared, the better the model fitting effect. However, our R^2 is only 0.29, which is relatively small. That shows that our regression model does not fit well. Another indicator, MSE, is called Mean Squared Error. It uses the difference between the predicted and actual values to judge the pros and cons of the model, where the smaller the difference, the better the model. Moreover, the size of MSE for our model is 0.52, which means our model is not accurate enough.

Random Forest Regression

```
from sklearn.pipeline import make_pipeline
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
pipeline = make_pipeline(preprocessing.StandardScaler(), RandomForestRegressor(n_estimators= 100))
```

```
hyperparameters = { 'randomforestregressor__max_features' : [ 'auto', 'sqrt', 'log2' ],
                    'randomforestregressor__max_depth': [ None, 5, 3, 1]}
```

```
from sklearn.model_selection import GridSearchCV
mrl2 = GridSearchCV(pipeline, hyperparameters, cv= 5)
mrl2.fit(x_train, y_train)
```

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                       ('randomforestregressor',
                                        RandomForestRegressor())]),
             param_grid=[('randomforestregressor__max_depth': [None, 5, 3, 1],
                          'randomforestregressor__max_features': ['auto', 'sqrt',
                                                                    'log2'])])
```

```
# Predict x_test after using cross-validation
pred2 = mrl2.predict(x_test)
pred2
```

```
array([5.11, 5.13, 6. , ..., 5.06, 5.31, 5.26])
```

```
# Evaluate the Model
from sklearn.metrics import mean_squared_error, r2_score, precision_score
print("The R2 value of mrl2 is: ", r2_score(y_test, pred2))
print("The MSE of mrl2 is: ", mean_squared_error(y_test, pred2))
```

```
The R2 value of mrl2 is: 0.5504972566494917
The MSE of mrl2 is: 0.3342478461538462
```

To improve performance, we reflect on the reasons why the model may not fit well and be inaccurate. One of the significant reasons is that the precision of the imported wine data is not the same, so we need to normalize it first. Normalization assumes that all features are centered around zero and have roughly the same variance. To achieve this, we need a modeling pipeline. We first transform the data using `StandardScaler()` and then use the `RandomForestRegressor()` function to fit the model. At the same time, we need to use hyperparameters. It is a parameter whose value is set before starting the learning process, not parameter data obtained through training. Typically, hyperparameters need to be optimized. It selects a set of optimal hyperparameters for the learning machine to improve the performance and effectiveness of learning. After that, we employed cross-validation. It is a process of reliably estimating the performance of a method that builds a model by training and evaluating our model multiple times using the same method. Finally, we still use R^2 and MSE to judge the quality of our model. We can see that our R^2 increased from 29% to 55%. It means that the fit of our model has become higher. Moreover, the MSE dropped from 52% to 33%. It means that the smaller the average distance of points in the test dataset from the model, the more accurate our model will be.

Use the Random Forest Regression to Predict the Wine Quality:

```
x_pre=pd.DataFrame({"fixed acidity":[6.4,7.1],
                    "volatile acidity":[0.27,0.28],
                    "citric acid":[0.30,0.35],
                    "residual sugar":[1.6,3.5],
                    "chlorides":[0.040,0.028],
                    "free sulfur dioxide":[19.0,35.0],
                    "total sulfur dioxide":[86.0,91.0],
                    "density":[0.99089,0.99022],
                    "pH":[3.32,2.96],
                    "sulphates":[0.65,0.33],
                    "alcohol":[11.5,12.1]})

x_pre
```

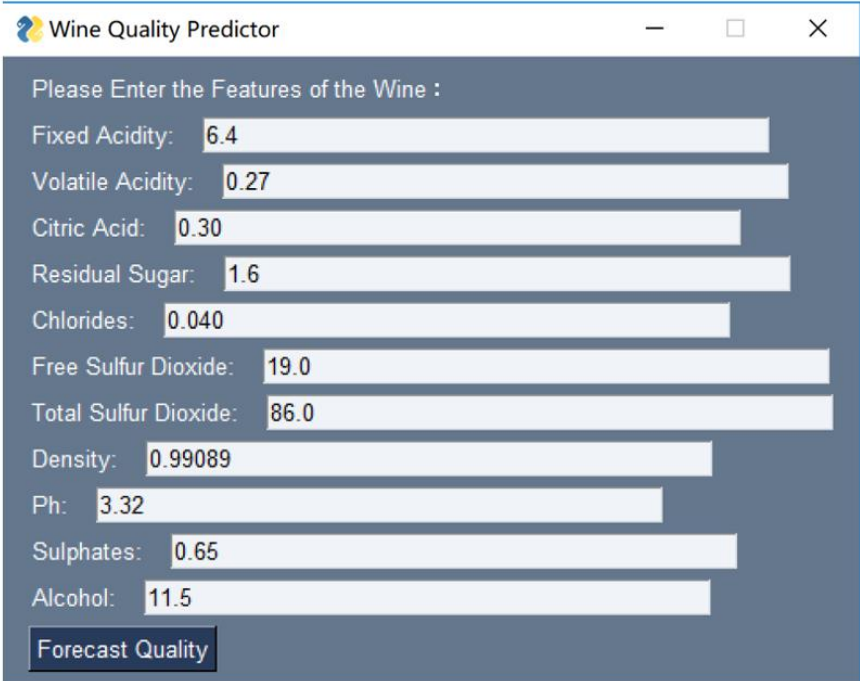
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	6.4	0.27	0.30	1.6	0.040	19.0	86.0	0.99089	3.32	0.65	11.5
1	7.1	0.28	0.35	3.5	0.028	35.0	91.0	0.99022	2.96	0.33	12.1

```
ml12.predict(x_pre)
array([6.15, 5.65])
```

Due to the improved fit and accuracy of the Random Forest Regression model, we use it to predict the wine quality. We fabricated two wines to verify the feasibility of the model. After

that, we created a data frame to store values for the two wines for each one of the predictor variables in our model. And the quality of the two wines was 6.15 and 5.65, respectively.

User Interface



The screenshot displays a graphical user interface titled "Wine Quality Predictor". It features a list of input fields for various wine attributes, each with a corresponding numerical value entered. The attributes and their values are: Fixed Acidity (6.4), Volatile Acidity (0.27), Citric Acid (0.30), Residual Sugar (1.6), Chlorides (0.040), Free Sulfur Dioxide (19.0), Total Sulfur Dioxide (86.0), Density (0.99089), Ph (3.32), Sulphates (0.65), and Alcohol (11.5). A "Forecast Quality" button is located at the bottom of the input section. To the right of the main window, a smaller popup window titled "F.." displays the prediction result: "The quality of the wine is 6", with an "OK" button below it.

Feature	Value
Fixed Acidity	6.4
Volatile Acidity	0.27
Citric Acid	0.30
Residual Sugar	1.6
Chlorides	0.040
Free Sulfur Dioxide	19.0
Total Sulfur Dioxide	86.0
Density	0.99089
Ph	3.32
Sulphates	0.65
Alcohol	11.5

In order to allow the user to correlate with the prediction results using the input, we designed a concise user interface with the help of the PySimpleGUI package. We built a popup window that allows people to enter various attributes of the wine and then click on "Forecast Quality" to predict the quality of the wine. This result was achieved using the Random Forest Regression model created earlier.