CS 521 Information Structures with Python
Wine Quality Analysis Report
Xiangxian Song, Weiyao Xu, Longyuan Tang
May 10, 2022

Introduction

The theme of our final project revolves around exploring wine quality. The project will involve two datasets with more than 1000 rows. They are related to red and white variants of the Portuguese "Vinho Verde" wine. We first visualized this data to understand further the different indicators and their relationship with the quality. After that, we use their attribute information to build a regression model to predict wine quality. Moreover, we create a classification model to understand further which of the two wines is of better quality. We also make a simple user interface to allow users to predict wine quality by entering various wine indicators. To understand our project in detail, compare this report with the code in the Jupyter notebook. This project will demonstrate the flexible use of Python packages such as SKlearn, Pandas, and Matplotlib.

Import the Dataset

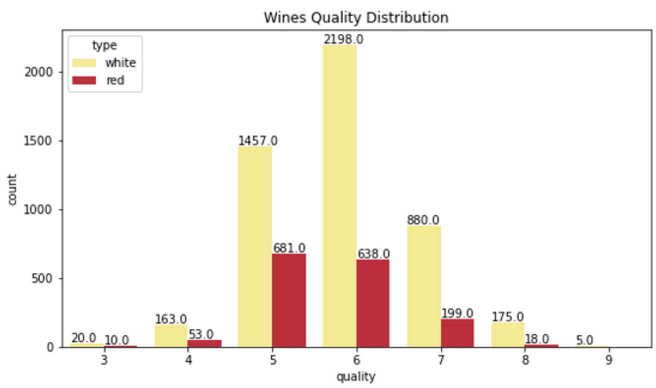
```
red = pd.read_csv('winequality-red.csv',sep=';')
white = pd.read_csv('winequality-white.csv',sep=';')
```

```
# store wine type as an attribute
red['type'] = 'red'
white['type'] = 'white'
# merge red and white wine datasets
wines = pd.concat([red, white])
# re-shuffle records to randomize data points
wines = wines.sample(frac = 1, random_state = 3).reset_index(drop = True)
```

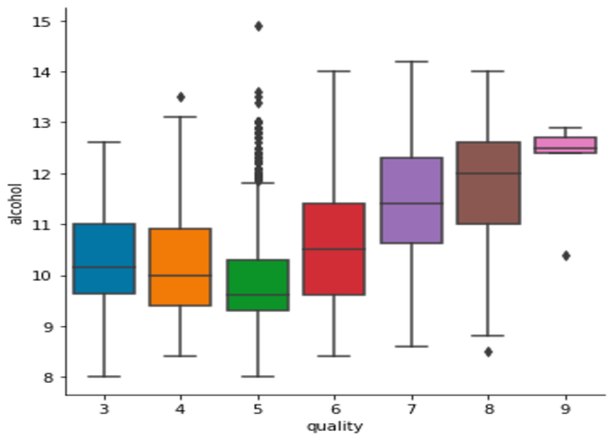
Download the “winequality-red.csv” and “winequality-white.csv” files from the UCI Machine Learning Repository. Then use the pd.read_cvs() function to import these data sets into our Python environment. In order to analyze and compare the characteristics of the two wines, we need to use the pd.concat() function to merge the two datasets together. Finally, all the data is shuffled. But the data for each sample is guaranteed to be the same as before, but the order has changed. After mixing, the index of the dataset is still sorted according to the normal order.

Dataset Exploration

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 the quality.



The 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.



To go further, after general checking the potential relationships between different data attributes through a 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.

We used heatmaps to examine potential relationships between different data attributes. Each square represents the correlation between two variables. The darker the color, the higher the negative correlation. The lighter the color, the higher the positive correlation.

To find suitable attributes to create a linear regression model, we need to compare the correlation of each attribute with wine quality. From this figure, we can know that wine quality has the highest correlation with alcohol. Other relation degrees are very low with each other, such as citric acid, free_sulfur_dioxide, sulphates, and pH. Quality also has a low negative correlation with density, volatile acidity, chlorides, total_sulfur_dioxide and residual_sugar.

The absolute values of correlations were considered: The columns pH, residual sugar, sulphates, total sulfur dioxide, free sulfur dioxide, fixed acidity, citric acid have very weak correlation (0.00 - 0.20); the columns chlorides, volatile acidity, density, and alcohol has weak correlation (0.20 - 0.40).

Split the Train and Test Data

```
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. 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. Moreover, we should ensure that the divided data should be random; otherwise, the model will be biased.

Fit the Linear Regression Model

```
# 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, pred1))
```

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

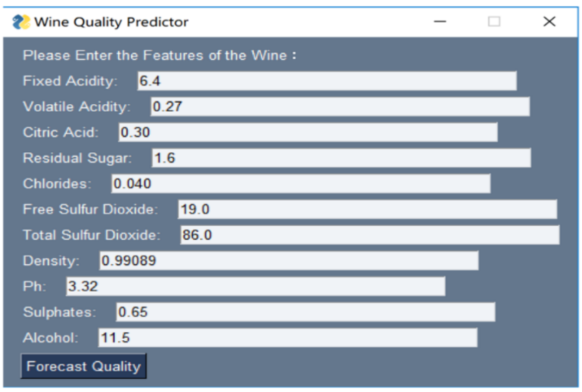
To further explore the relationship between wine quality and other variables, we will perform a Multiple Regression Analysis. Set wine quality as the dependent variable and other variables as independent variables. To achieve our goal, we use the linear_model.LinearRegression() function to build a regression 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.

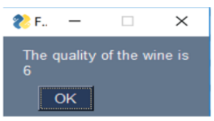
Random Forest Regression Model

```
# 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.





User Interface

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.

Classification

Since the column that we should predict is from 1 to 10. Instead of using a regression model, our team decided to use a classification model and pick which one is better. There are two models we used, random forest classification model and support vector machine model. First, since the type_wine is a categorical variable, the computer cannot recognize string type. We need a dummified type_wine column to 0 and 1 where white wine is 0 and red wine is 1. We do not need to scale the data as we do in a regression session, because a classification model cannot build a linear model which needs everything in the same standard deviation.

```
wines.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	type_white
0	5.9	0.180	0.28	1.0	0.037	24.0	88.0	0.99094	3.29	0.55	10.65	7	1
1	10.2	0.670	0.39	1.9	0.054	6.0	17.0	0.99760	3.17	0.47	10.00	5	0
2	8.4	0.715	0.20	2.4	0.076	10.0	38.0	0.99735	3.31	0.64	9.40	5	0
3	6.8	0.370	0.51	11.8	0.044	62.0	163.0	0.99760	3.19	0.44	8.80	5	1
4	8.9	0.750	0.14	2.5	0.086	9.0	30.0	0.99824	3.34	0.64	10.50	5	0

Supervised experiment is necessary for classification models, because we need to use a grid search CV function to list all the parameters and select the best one from it, we need to use supervised experiment to know whether our data is overfitting to the training dataset or not. However, an overfitting model is not a bad model for random forest, if the predicted result is good, it does not matter if the model is overfitting. After listing all the parameters, the best pattern is max_depth = 11, max_features=9, n_estimators=150. The accuracy of the model is 63%. While splitting the test and training data, we decide to use 0.45 because the data while wine quality less than 3 or greater than 8 are few. If defining only 30% data as train data, it cannot train any data marked as quality of 1, which will influence the result of the model.

```
param_grid = {
    'n_estimators':[100,150,200],
    'max_depth':[5,10,15],
    'max_features':[7,9,11],
    'min_samples_leaf':[10,15,20],
}
grid = GridSearchCV(estimator=clf,param_grid=param_grid,cv=5,verbose=3)
grid.fit(X_train,y_train)
```

```
[CV 4/5] END max_depth=5, max_features=7, min_samples_leaf=10, n_estimators=100;, score=0.549 total time=   0.8s
[CV 5/5] END max_depth=5, max_features=7, min_samples_leaf=10, n_estimators=100;, score=0.562 total time=   0.8s
[CV 1/5] END max_depth=5, max_features=7, min_samples_leaf=10, n_estimators=150;, score=0.562 total time=   1.3s
```

```
print(confusion_matrix(y_test,pred_y),'\n',confusion_matrix(y_train,pred_y1))
print()
print(classification_report(y_test,pred_y),classification_report(y_train,pred_y1))
```

```
[[  0   0   6   7   0   0   0]
 [  0   3  54  27   0   0   0]
 [  0   2 575 248  10   0   0]
 [  0   0 220 838  79   0   0]
 [  0   0  21 228 208   2   0]
 [  0   0   0  36  19  13   0]
 [  0   0   0   1   2   0   0]]
 [[  14    0    2    1    0    0    0]
 [   0   91   31   10    0    0    0]
 [   0    0 1217   83    3    0    0]
 [   0    0   83 1613    3    0    0]
 [   0    0    6   74  540    0    0]
 [   0    0    1   28   12   84    0]
 [   0    0    0    1    1    0    0]]

              precision    recall  f1-score   support

           3       0.00      0.00      0.00        13
           4       0.60      0.04      0.07        84
           5       0.66      0.69      0.67       835
           6       0.61      0.74      0.66      1137
           7       0.65      0.45      0.54       459
           8       0.87      0.19      0.31        68
           9       0.00      0.00      0.00         3

    accuracy                           0.63      2599
   macro avg       0.48      0.30      0.32      2599
weighted avg       0.63      0.63      0.61      2599
               precision    recall  f1-score   support

           3       1.00      0.82      0.90        17
           4       1.00      0.69      0.82       132
           5       0.91      0.93      0.92      1303
           6       0.89      0.95      0.92      1699
           7       0.97      0.87      0.92       620
           8       1.00      0.67      0.80       125
           9       0.00      0.00      0.00         2

    accuracy                           0.91      3898
   macro avg       0.82      0.71      0.75      3898
weighted avg       0.92      0.91      0.91      3898
```

The accuracy is pretty good since it is a real dataset, it is hard to achieve an accuracy of 0.70 or more. In order to make sure the model is the best classification model for this dataset, the other model, the support vector machine, is applied to the same dataset. It also applied a grid search CV function to find the best pattern by changing C or gamma value. As a result, the best model has an accuracy of 0.54 and the best pattern is when C is 0.1 and gamma is 0.001.

```
In [117]: grid.fit(X_train,y_train)
          [CV 1/5] END ...............C=1000, gamma=0.01;, score=0.578 total time=   6.2s
          [CV 2/5] END ...............C=1000, gamma=0.01;, score=0.533 total time=   6.0s
          [CV 3/5] END ...............C=1000, gamma=0.01;, score=0.523 total time=   6.5s
          [CV 4/5] END ...............C=1000, gamma=0.01;, score=0.544 total time=   5.9s
          [CV 5/5] END ...............C=1000, gamma=0.01;, score=0.540 total time=   6.5s
          [CV 1/5] END ..............C=1000, gamma=0.001;, score=0.569 total time=   6.9s
          [CV 2/5] END ..............C=1000, gamma=0.001;, score=0.549 total time=   6.8s
          [CV 3/5] END ..............C=1000, gamma=0.001;, score=0.546 total time=   7.8s
          [CV 4/5] END ..............C=1000, gamma=0.001;, score=0.573 total time=   7.0s
          [CV 5/5] END ..............C=1000, gamma=0.001;, score=0.534 total time=   6.8s
          [CV 1/5] END .............C=1000, gamma=0.0001;, score=0.546 total time=   7.1s
          [CV 2/5] END .............C=1000, gamma=0.0001;, score=0.544 total time=   7.3s
          [CV 3/5] END .............C=1000, gamma=0.0001;, score=0.559 total time=  10.0s
          [CV 4/5] END .............C=1000, gamma=0.0001;, score=0.547 total time=   7.0s
          [CV 5/5] END .............C=1000, gamma=0.0001;, score=0.547 total time=   7.3s

Out[117]: GridSearchCV(estimator=SVC(),
                       param_grid={'C': [0.1, 1, 10, 100, 1000],
                                   'gamma': [1, 0.1, 0.01, 0.001, 0.0001]},
                       verbose=3)
```

```
print(confusion_matrix(y_test,pred_y),'\n',confusion_matrix(y_train,pred_y1))
print()
print(classification_report(y_test,pred_y),classification_report(y_train,pred_y1))
```

```
[[   1    2    7    3    0    0    0]
 [   1    4   56   17    5    1    0]
 [   1    9  554  247   22    2    0]
 [   0   10  317  722   84    3    1]
 [   0    1   44  283  131    0    0]
 [   0    1    7   36   20    4    0]
 [   0    0    0    2    1    0    0]]
 [[  12    0    4    1    0    0    0]
 [   0   33   66   32    1    0    0]
 [   1    2  951  331   17    1    0]
 [   0    4  334 1269   91    1    0]
 [   1    3   32  347  236    1    0]
 [   0    0   10   56   36   23    0]
 [   0    0    0    0    0    0    2]]

              precision    recall  f1-score   support

           3       0.33      0.08      0.12        13
           4       0.15      0.05      0.07        84
           5       0.56      0.66      0.61       835
           6       0.55      0.64      0.59      1137
           7       0.50      0.29      0.36       459
           8       0.40      0.06      0.10        68
           9       0.00      0.00      0.00         3

    accuracy                           0.54      2599
   macro avg       0.36      0.25      0.27      2599
weighted avg       0.53      0.54      0.52      2599
               precision    recall  f1-score   support

           3       0.86      0.71      0.77        17
           4       0.79      0.25      0.38       132
           5       0.68      0.73      0.70      1303
           6       0.62      0.75      0.68      1699
           7       0.62      0.38      0.47       620
           8       0.88      0.18      0.30       125
           9       1.00      1.00      1.00         2

    accuracy                           0.65      3898
   macro avg       0.78      0.57      0.62      3898
weighted avg       0.66      0.65      0.63      3898
```

From those two models we infer that the random forest model is better for classification in this dataset. However, the flaw is obvious, the confusion matrix shows there is still no data where wine quality is equal to 1 in the test group. Thus, we do not know the accuracy when using this model to estimate the low-quality wines.

Conclusion and Suggestion

By comparing the results from different models, our team concluded that a random forest regression model is better for the entire dataset. Firstly, a classification model has uncertainty due lack of data in low quality wines. A regression model can predict value through some linear or nonlinear relationship which has a better prediction on those lacking data.