Hongdi Li

IST-652

Research on wine datasets

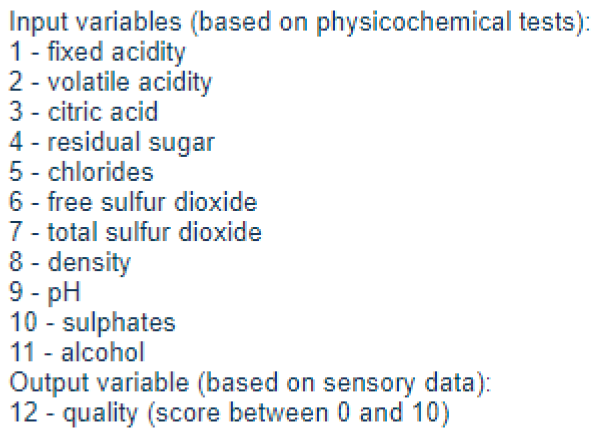
https://youtu.be/NGy8SJfzvmg

Abstract

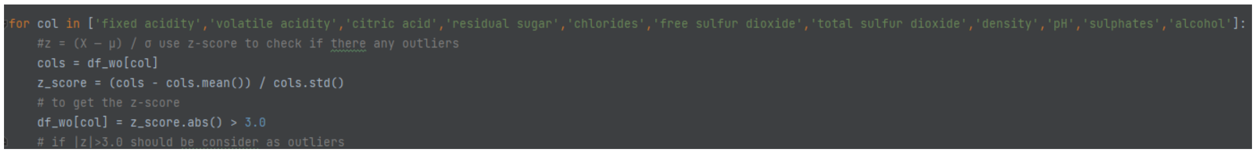
The data set I studied comes from the "Wine Quality Data Set" by P. Cortez, A. Cerdeira, etc. Which located in the UCI database. This data mainly records the chemical composition and wine quality in red and white wines (from 1 to 10). In my knowledge of external sources, such as Kerri-Ann Jennings' article "Red Wine vs White Wine: Which Is Healthier?" clearly states "The main difference between white and red wine has to do with the color of the grapes used. It also has to do with whether the grape juice is fermented with or without the grape skin. (Jennings)" So I don't think there is much difference in composition between red wine and white wine, I want to wine building classification models to predict wine quality by the different ingredients contained.

Preprocessing

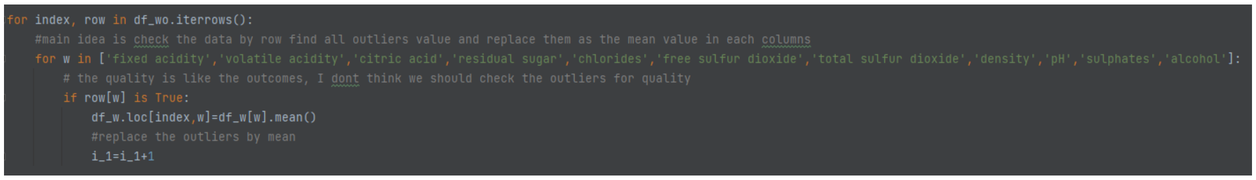
I first do file preprocessing on the dataset. By looking at the file, it is found that the two datasets have the same labels, namely fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. To help me distinguish red wine and white wine data, I added a new column to the two dataframes - "'wine_type'. Use numbers 1 and 0 to distinguish red wine from white wine, 1 is white wine, 0 is red wine .


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)

Then I found no missing values in both red wine and white wine by .isna().sum().sum() function. Find outliers in the dataframe except for the 'quality' column by z-score (Z = (X-μ)/σ). I treat all data with |z|>3 as outliers, flag those data, and replace them with the mean of the column they are in.

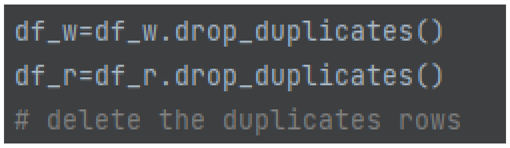


```python
for col in ['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','free sulfur dioxide','total sulfur dioxide','density','pH','sulphates','alcohol']:
    #z = (X — μ) / σ use z-score to check if there any outliers
    cols = df_wo[col]
    z_score = (cols - cols.mean()) / cols.std()
    # to get the z-score
    df_wo[col] = z_score.abs() > 3.0
    # if |z|>3.0 should be consider as outliers
```



```python
for index, row in df_wo.iterrows():
    #main idea is check the data by row find all outliers value and replace them as the mean value in each columns
    for w in ['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','free sulfur dioxide','total sulfur dioxide','density','pH','sulphates','alcohol']:
        # the quality is like the outcomes, I dont think we should check the outliers for quality
        if row[w] is True:
            df_w.loc[index,w]=df_w[w].mean()
            #replace the outliers by mean
            i_1=i_1+1
```

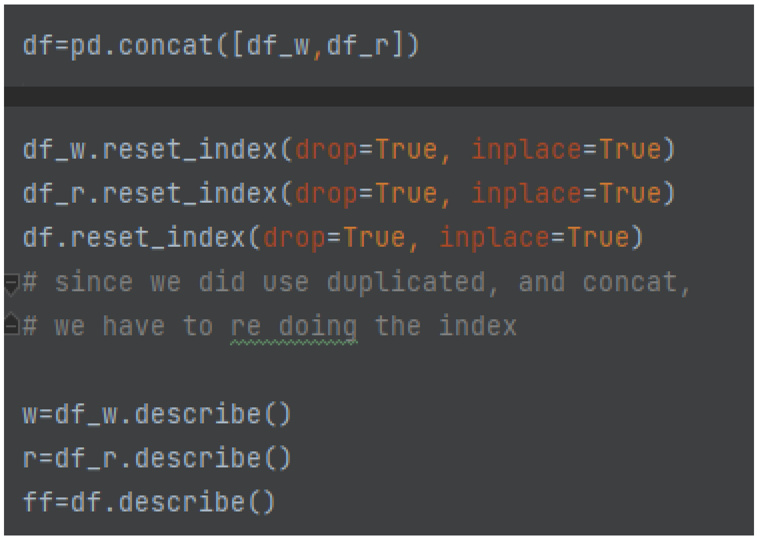
I think the quality of wine is mainly made by the taster through different factors such as appearance, smell, taste and so on. Since everyone's definition of wine quality is different, so long as the quality is within 1-10 is reasonable. So, I don't think an outlier analysis for the 'quality' column is needed. Finally, find duplicate rows through the duplicated() function to remove duplicate rows in each dataframe.



```python
df_w=df_w.drop_duplicates()
df_r=df_r.drop_duplicates()
# delete the duplicates rows
```

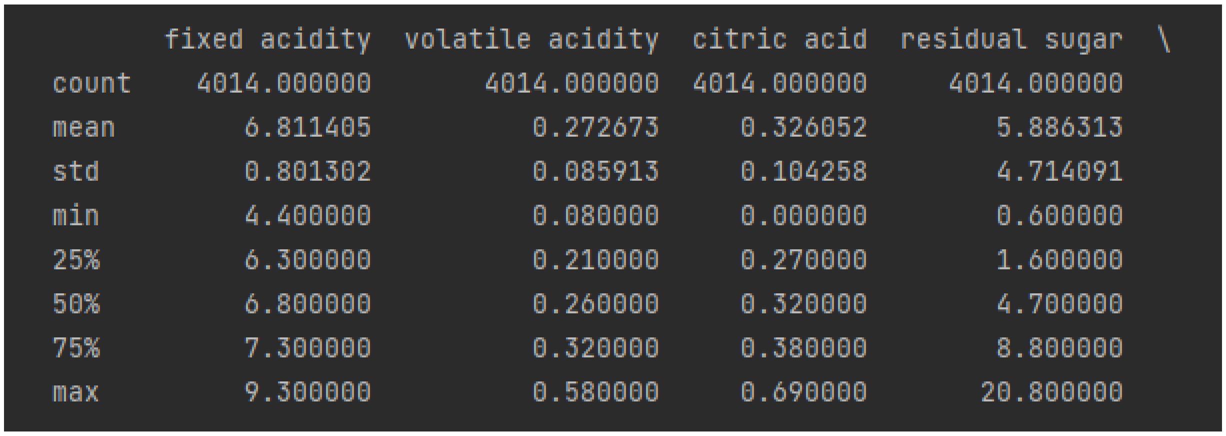
Check the data

I “merge” the data of red wine and white wine to get a new dataset df.

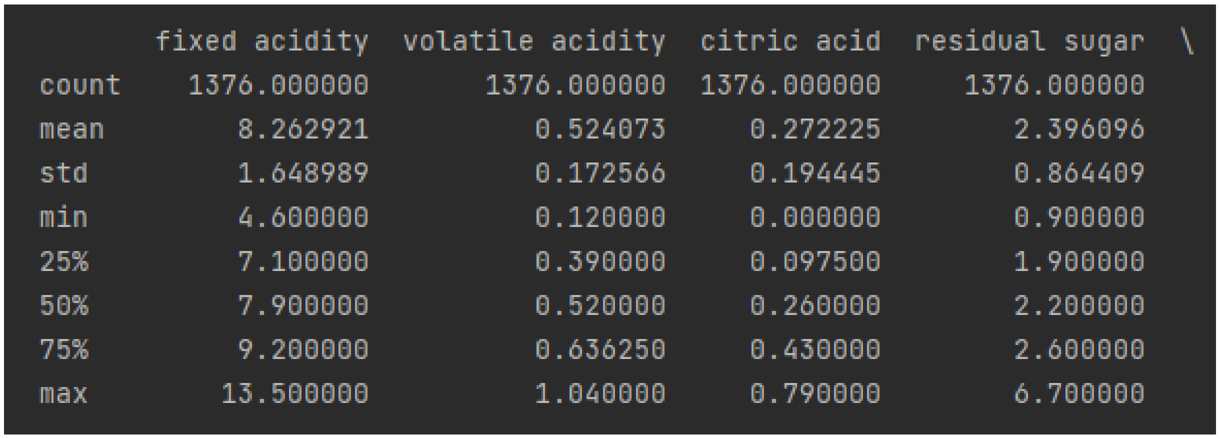


```
df=pd.concat([df_w,df_r])

df_w.reset_index(drop=True, inplace=True)
df_r.reset_index(drop=True, inplace=True)
df.reset_index(drop=True, inplace=True)
# since we did use duplicated, and concat,
# we have to re doing the index

w=df_w.describe()
r=df_r.describe()
ff=df.describe()
```

At this point I have 3 dataframes. Reorder the indices in each dataframe and view the information in each dataframe. At this point there are 4014 rows for the red wine dataframe and 1376 rows for the white wine dataframe. And the new df has 5390 rows. Looking at the data information for each column, I found that white wines are roughly the same as red wines, except for individual components such as residual sugar and volatile acidity that are slightly different from red wines.



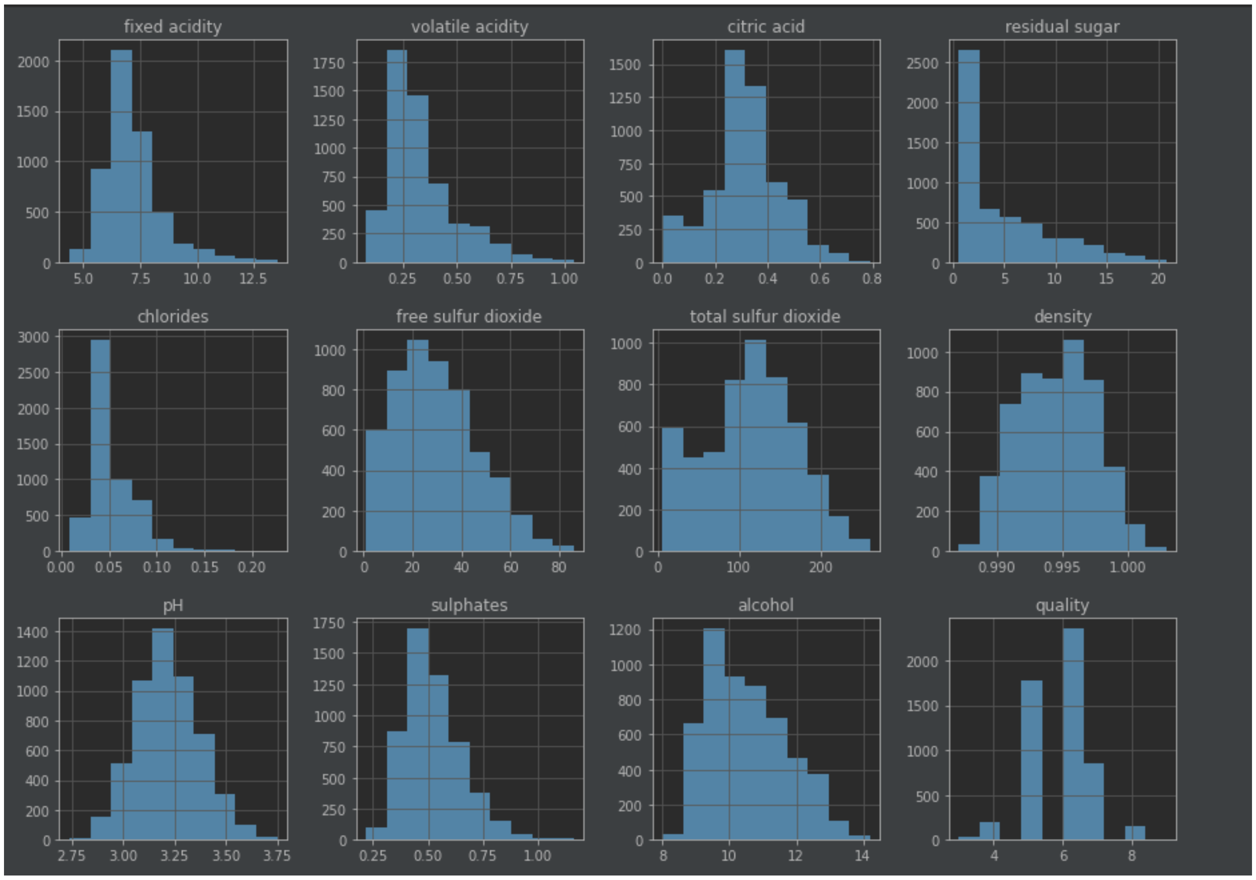
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4014.000000       4014.000000  4014.000000     4014.000000
mean        6.811405          0.272673     0.326052        5.886313
std         0.801302          0.085913     0.104258        4.714091
min         4.400000          0.080000     0.000000        0.600000
25%         6.300000          0.210000     0.270000        1.600000
50%         6.800000          0.260000     0.320000        4.700000
75%         7.300000          0.320000     0.380000        8.800000
max         9.300000          0.580000     0.690000       20.800000
```



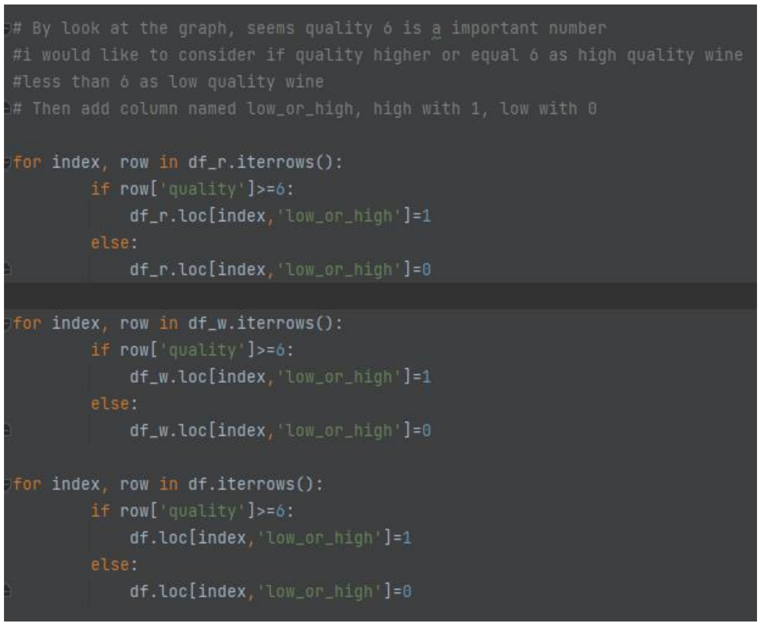
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1376.000000       1376.000000  1376.000000     1376.000000
mean        8.262921          0.524073     0.272225        2.396096
std         1.648989          0.172566     0.194445        0.864409
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.097500        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.636250     0.430000        2.600000
max        13.500000          1.040000     0.790000        6.700000
```

(These are part information of the red/white wine dataframe. Top is whilte, below is red)

By plotting each label in the three dataframes separately. I found that most of the images had a normal distribution shifted to the left, and a few images such as pH had an almost standard normal distribution. For the 'quality' classification, the data is mainly concentrated in 5-7.



At first I wanted to classify the quality of the wine into three categories: good, medium and bad. But in this way, the quality of most wines should be classified as mid-range wines. So I set 6 as the dividing line, wines with a quality less than 6 are considered inferior wines, and wines with a quality greater than 6 are considered superior wines. Similarly, I categorize the

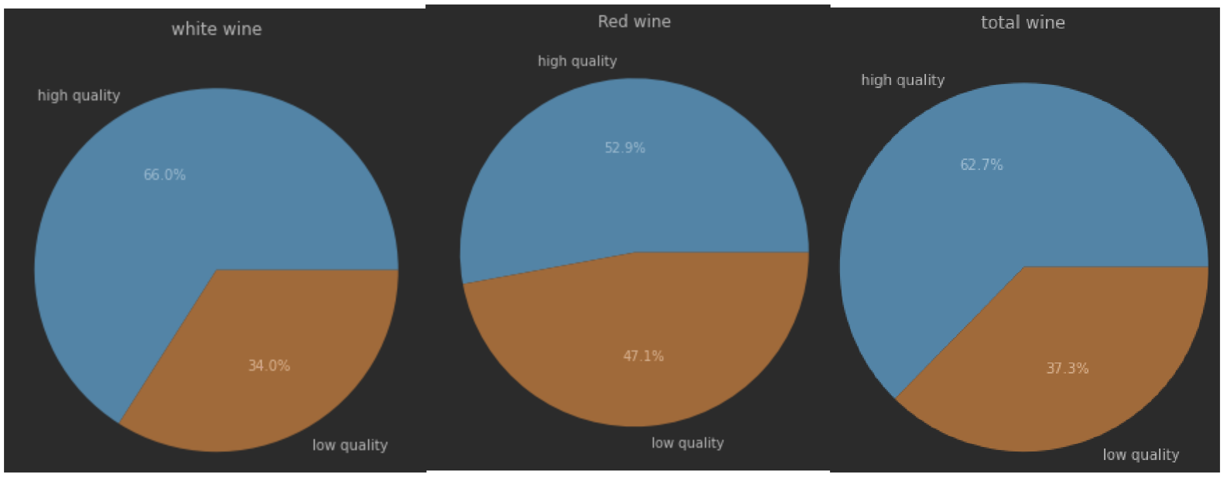


```
# By look at the graph, seems quality 6 is a important number
#i would like to consider if quality higher or equal 6 as high quality wine
#less than 6 as low quality wine
# Then add column named low_or_high, high with 1, low with 0

for index, row in df_r.iterrows():
        if row['quality']>=6:
            df_r.loc[index,'low_or_high']=1
        else:
            df_r.loc[index,'low_or_high']=0

for index, row in df_w.iterrows():
        if row['quality']>=6:
            df_w.loc[index,'low_or_high']=1
        else:
            df_w.loc[index,'low_or_high']=0

for index, row in df.iterrows():
        if row['quality']>=6:
            df.loc[index,'low_or_high']=1
        else:
            df.loc[index,'low_or_high']=0
```

'quality' row by row in the dataframe, and store the results in a new column "low_or_high", where 1 means good wine and 0 means bad wine.

Visualization

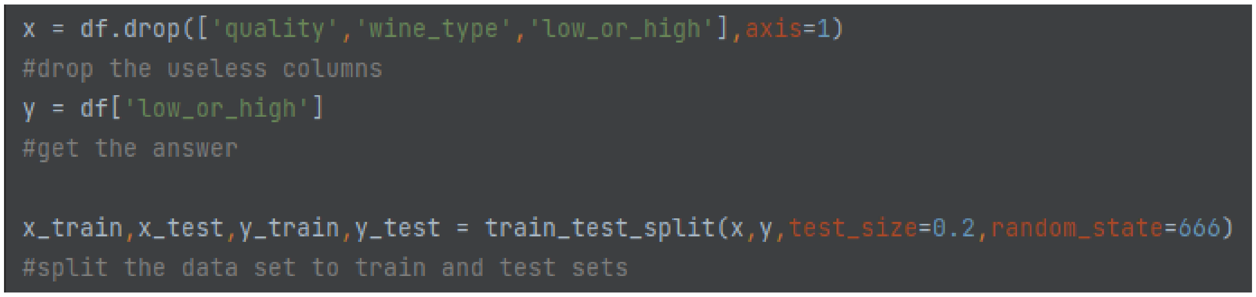
Then draw a pie chart by calling matplotlib. I want to visualize the weights of good wine and bad wine in the dataset through a pie chart. By drawing, it is found that the proportion of good wines is much larger than that of bad wines for the red wine dataset, but in the white wine



dataset, the proportion of good and bad wines is almost the same. In the overall dataset, there are more good wines than bad wines.

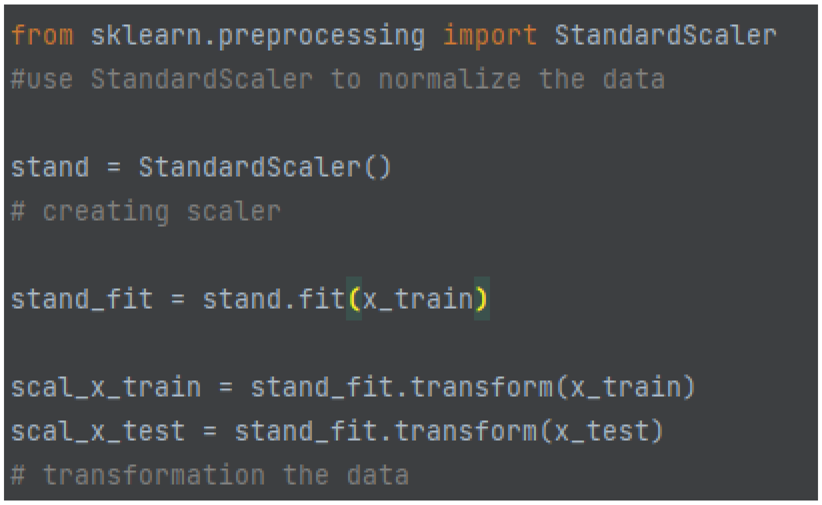
Machine learning

I think to classify data one should use decision tree. Considering that I have a lot of classifications in my data. I think using a random forest classifier is a good option. Import RandomForestClassifier via sklearn library. Define x, y. I want the program to observe all categories except 'quality', 'wine_type', 'low_or_high'. Then define y to be 'low_or_high'. In this way, the model will analyze whether the wine is a good wine or a bad wine based on all the components of the wine. Finally, the data is divided into 20% test set and 80% learning set.



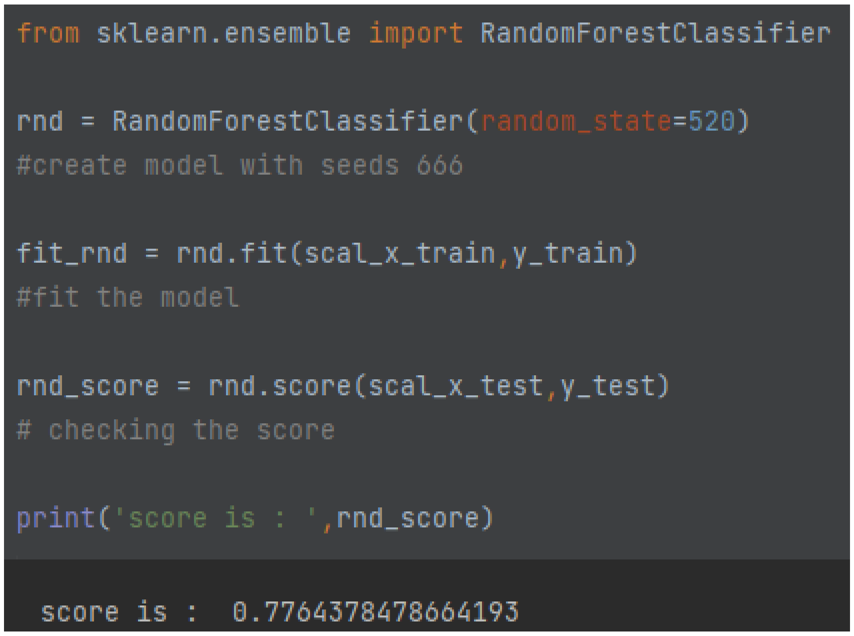
```
x = df.drop(['quality','wine_type','low_or_high'],axis=1)
#drop the useless columns
y = df['low_or_high']
#get the answer

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=666)
#split the data set to train and test sets
```

Considering the large variance in the size of the data in the dataset, I think model normalization is needed to make the model learn more efficiently. So I use StandardScaler() function to standardize the model.



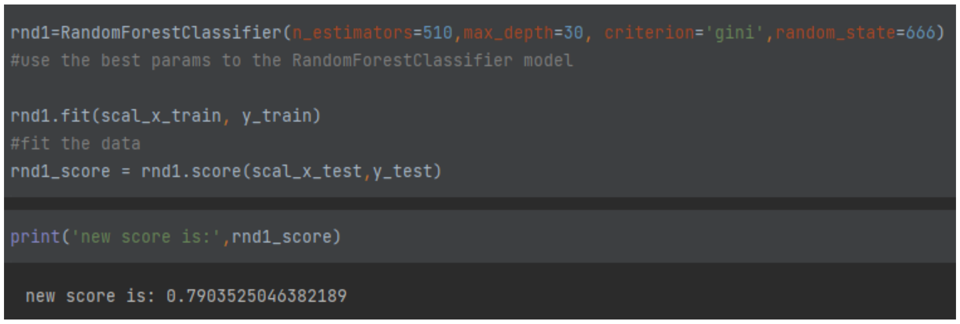
```
from sklearn.preprocessing import StandardScaler
#use StandardScaler to normalize the data

stand = StandardScaler()
# creating scaler

stand_fit = stand.fit(x_train)

scal_x_train = stand_fit.transform(x_train)
scal_x_test = stand_fit.transform(x_test)
# transformation the data
```

View the learning results by using RandomForestClassifier(). The final result was 77.6%. This means that my model is 77.6% accurate in predicting the quality of wine.



```
from sklearn.ensemble import RandomForestClassifier

rnd = RandomForestClassifier(random_state=520)
#create model with seeds 666

fit_rnd = rnd.fit(scal_x_train,y_train)
#fit the model

rnd_score = rnd.score(scal_x_test,y_test)
# checking the score

print('score is : ',rnd_score)

 score is :  0.7764378478664193
```

Then adjust the parameters through GridSearchCV to enhance the accuracy of the model. For this model, I mainly adjust the parameters of "n_estimators", "max_depth", and "criterion". Through the GridSearchCV() operation, the optimal "n_estimators=510, max_depth=30,



```
rnd1=RandomForestClassifier(n_estimators=510,max_depth=30, criterion='gini',random_state=666)
#use the best params to the RandomForestClassifier model

rnd1.fit(scal_x_train, y_train)
#fit the data
rnd1_score = rnd1.score(scal_x_test,y_test)

print('new score is:',rnd1_score)

 new score is: 0.7903525046382189
```

criterion='gini'" is finally obtained, and the model is set to the optimal parameters to relearn, and the final accuracy rate is 0.79, which is a certain improvement compared to the original data of 0.77.

Future improvement

I think I have too little data on red wines in my total dataset, less than twice as much data on white wines as red wines. If I had more data on red wine it seemed to be able to know if the type of wine had any influence on the quality of the wine. On the other hand, wine quality is manually scored by different sommeliers. So I don't think it seems very wise to remove duplicate rows. I think duplicate rows should be kept, identical rows of data are more likely to define wine standards better. In the process of parameter adjustment, the running speed is very slow due to the poor performance of my computer. Otherwise, I think that the range of parameter adjustment can be increased to enhance the accuracy of the model. If there is a chance in the future, I should further adjust the parameters to make the model more accurate. On the other hand, I would also consider building a classification model for predict wine type.

Citation

Jennings, Kerri-Ann. “Red Wine vs White Wine: Which Is Healthier?” *Healthline*, Healthline Media, 17 Feb. 2017, https://www.healthline.com/nutrition/red-vs-white-wine.

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.