Wine Quality Classification

IMLP439/蘇家平

2025/04

Wine Quality Dataset

- This datasets is related to red variants of the Portuguese “Vinho Verde” wine. The dataset describes the amount of various chemicals present in wine and their effect on it‘s quality. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones). Task is to predict the quality of wine using the given data.
- https://www.kaggle.com/datasets/yasserh/wine-quality-dataset/data

0. Import functions

```
# For data analysis
import pandas as pd
import numpy as np

# For figure
import seaborn as sns
import matplotlib.pyplot as plt

# For Scikit-learn function
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score

from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import accuracy_score

from utility import plot_confusion_matrix,plot_decision_regions,testcase_report
```

1. Collect and clean data (1/3)

This data frame contains:

- 11 columns of chemical items
- 1 column of quality score (3~8)
- 1 column of wine ID

```
# Input dataset
data = pd.read_csv('WineQT.csv')
data
```

```
# check data range
for i in data[:0]:
    print(i, 'range from', data[i].min(), 'to', data[i].max())
```

```
固定酸度 fixed acidity range from 4.6 to 15.9
揮發性酸度 volatile acidity range from 0.12 to 1.58
檸檬酸 citric acid range from 0.0 to 1.0
殘糖 residual sugar range from 0.9 to 15.5
氯化物 chlorides range from 0.012 to 0.611
遊離二氧化硫 free sulfur dioxide range from 1.0 to 68.0
總二氧化硫 total sulfur dioxide range from 6.0 to 289.0
密度 density range from 0.99007 to 1.00369
pH值 pH range from 2.74 to 4.01
硫酸鹽 sulphates range from 0.33 to 2.0
酒精 alcohol range from 8.4 to 14.9
quality range from 3 to 8
Id range from 0 to 1597
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	0
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	2
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	3
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6	1592
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6	1593
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5	1594
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6	1595
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5	1597

1143 rows × 13 columns

1. Collect and clean data (2/3)

This dataset is clear.

```
# Check this dataset's quality
print(pd.isnull(data).sum())
```

```
fixed acidity         0
volatile acidity      0
citric acid           0
residual sugar        0
chlorides             0
free sulfur dioxide   0
total sulfur dioxide  0
density               0
pH                    0
sulphates             0
alcohol               0
quality               0
Id                    0
dtype: int64
```

However, the feature 'Id' is not useful, and we will drop it.

```
data.drop(labels=['Id'], axis=1, inplace=True)
data
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5

1143 rows × 12 columns

1. Collect and clean data (3/3)

Standardize all features

```
scaler = StandardScaler()

# quality is answer, don't need to standardize
scaler.fit(data.drop('quality', axis=1))
scaler_features = scaler.transform(data.drop('quality', axis=1))

df_data = pd.DataFrame(scaler_features, columns=data.columns[:-1])
df_data
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	-0.521580	0.939332	-1.365027	-0.466421	-0.231395	-0.450467	-0.363610	0.555854	1.270695	-0.573658	-0.963382
1	-0.292593	1.941813	-1.365027	0.050060	0.234247	0.915920	0.643477	0.036165	-0.708928	0.130881	-0.593601
2	-0.292593	1.273492	-1.161568	-0.171289	0.107253	-0.060071	0.246745	0.140103	-0.325775	-0.045254	-0.593601
3	1.653789	-1.399789	1.483400	-0.466421	-0.252560	0.135127	0.429852	0.659792	-0.964363	-0.456235	-0.593601
4	-0.521580	0.939332	-1.365027	-0.466421	-0.231395	-0.450467	-0.363610	0.555854	1.270695	-0.573658	-0.963382
...	...	...	...	...	...	...	...	...	...	...	...
1138	-1.151292	-0.118842	-0.703785	-0.171289	-0.231395	1.306316	-0.180503	-0.514707	0.695966	0.541862	0.515741
1139	-0.865059	0.493785	-0.958109	-0.466421	-0.400719	1.208717	-0.241539	-0.114545	0.695966	0.952843	-0.870937
1140	-1.208538	0.382399	-0.958109	-0.392638	0.064922	1.599113	-0.058432	-0.951246	0.887542	-0.456235	0.053515
1141	-1.380278	0.103932	-0.856379	-0.245072	-0.527712	2.282306	0.155192	-0.836914	1.334554	0.600574	0.700632
1142	-1.380278	0.633019	-0.754650	-0.392638	-0.252560	1.599113	-0.058432	-0.655023	1.653848	0.307016	-0.223820

1143 rows × 11 columns

2. Train through various models

Split 70% training and 30% testing data

```
# Split 70% training and 30% testing data
X = df_data
y = data['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
print('train data quantity:', X_train.shape, ';', 'test data quantity:', X_test.shape)
```

```
train data quantity: (800, 11) ; test data quantity: (343, 11)
```

2. Train through various models (1/2)

Calculate by different model and descide the best model

(1) Logistic Regression;
(2) Decision Tree;
(3) Support Vector Classifier;
(4) K-Nearest Neighbors;
(5) RandomForest;
(6) AdaBoost

```python
# Logistic Regression
LogReg_clf = LogisticRegression()
LogReg_clf.fit(X_train, y_train)
LogReg_pred = LogReg_clf.predict(X_test)
acc_LogReg = accuracy_score(y_test, LogReg_pred)

# Decision Tree
DTree_clf = DecisionTreeClassifier()
DTree_clf.fit(X_train, y_train)
DTree_pred = DTree_clf.predict(X_test)
acc_DTree = accuracy_score(y_test, DTree_pred)

# Support Vector Classifier
SVC_clf = SVC()
SVC_clf.fit(X_train, y_train)
SVC_pred = SVC_clf.predict(X_test)
acc_SVC = accuracy_score(y_test, SVC_pred)
```

```python
# K-Nearest Neighbors
knn_error_rate = []
knn = KNeighborsClassifier(n_neighbors=100)
knn.fit(X_train, y_train)
knn_error_rate = np.mean(knn.predict(X_test) != y_test)

# RandomForest
random_forest = RandomForestClassifier(criterion='entropy', n_estimators=50, random_state=100, n_jobs=20)
random_forest.fit(X_train, y_train)
random_forest_error_rate = random_forest.score(X_test, y_test)

# AdaBoost
AdaBoost = AdaBoostClassifier(n_estimators=100)
AdaBoost.fit(X_train, y_train)
adaBoost_error_rate = AdaBoost.score(X_test, y_test)
```

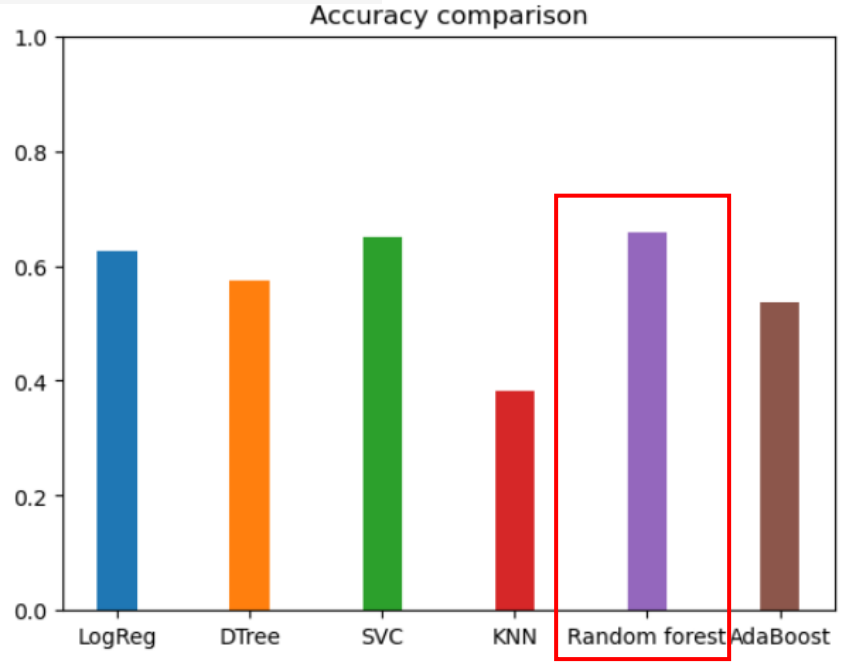
2. Train through various models (2/2)

Calculate accuracy

```
# Compare accuracy
print('Accuracy summary')
print('LogReg: ', acc_LogReg)
print('DTree: ', acc_DTree)
print('SVC: ', acc_SVC)
print('KNN: ', knn_error_rate)
print('Random forest: ', random_forest_error_rate)
print('AdaBoost: ', adaBoost_error_rate)

# Plot bar chart
x = ['LogReg', 'DTree', 'SVC', 'KNN', 'Random forest', 'AdaBoost']
y = [acc_LogReg, acc_DTree, acc_SVC, knn_error_rate, random_forest_error_rate, adaBoost_error_rate]
color = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']
plt.bar(x, y, width=0.3, color=color)
plt.ylim([0, 1])
plt.title('Accuracy comparison')
plt.show()
```

```
Accuracy summary
LogReg:  0.6268221574344023
DTree:  0.5743440233236151
SVC:  0.6501457725947521
KNN:  0.3819241982507289
Random forest:  0.6647230320699709
AdaBoost:  0.5364431486880467
```



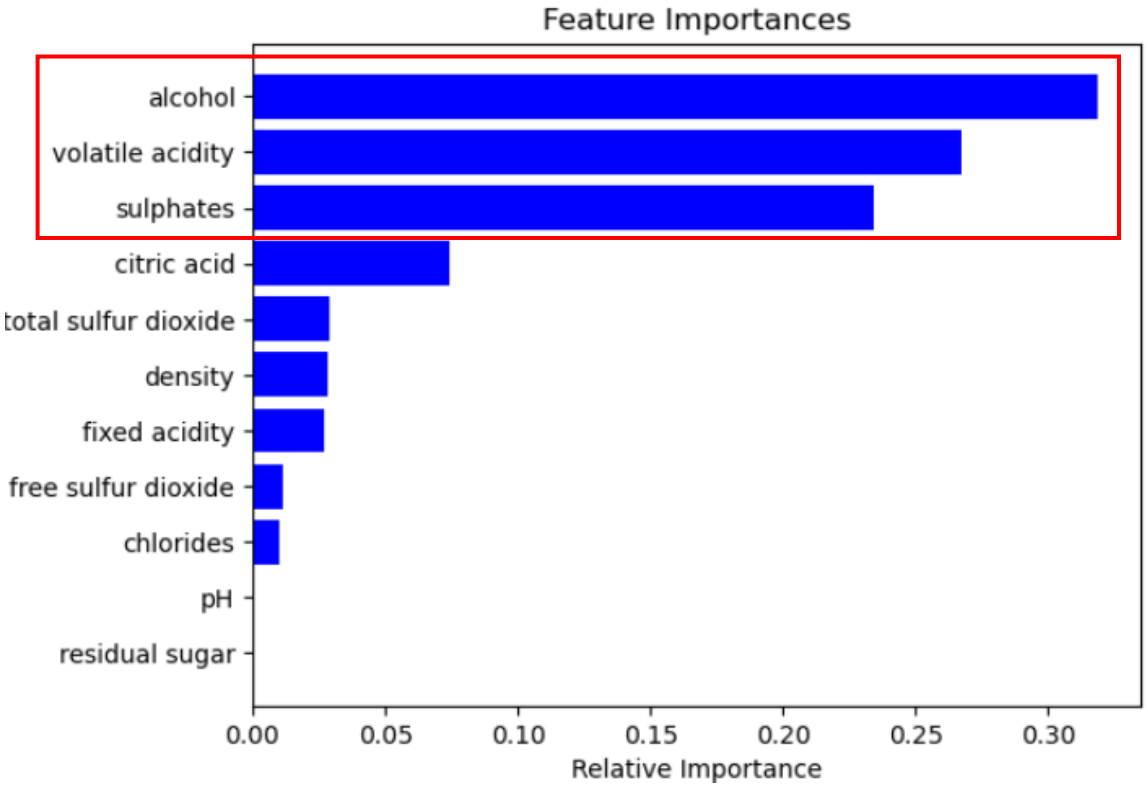
'Random forest' is the best model for this dataset, error rate = 0.66

3. Analysis the top 3 important items (1/2)

```
# Calculate the features' weight
importances = AdaBoost.feature_importances_
indices = np.argsort(importances)  #np.argsort :Returns the indices that would sort an array.
features = X_train.keys()
print(importances[indices])
print(features[indices])

# Plot bar chart
plt.figure(1)
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), features[indices])
plt.xlabel('Relative Importance')
plt.show()
```

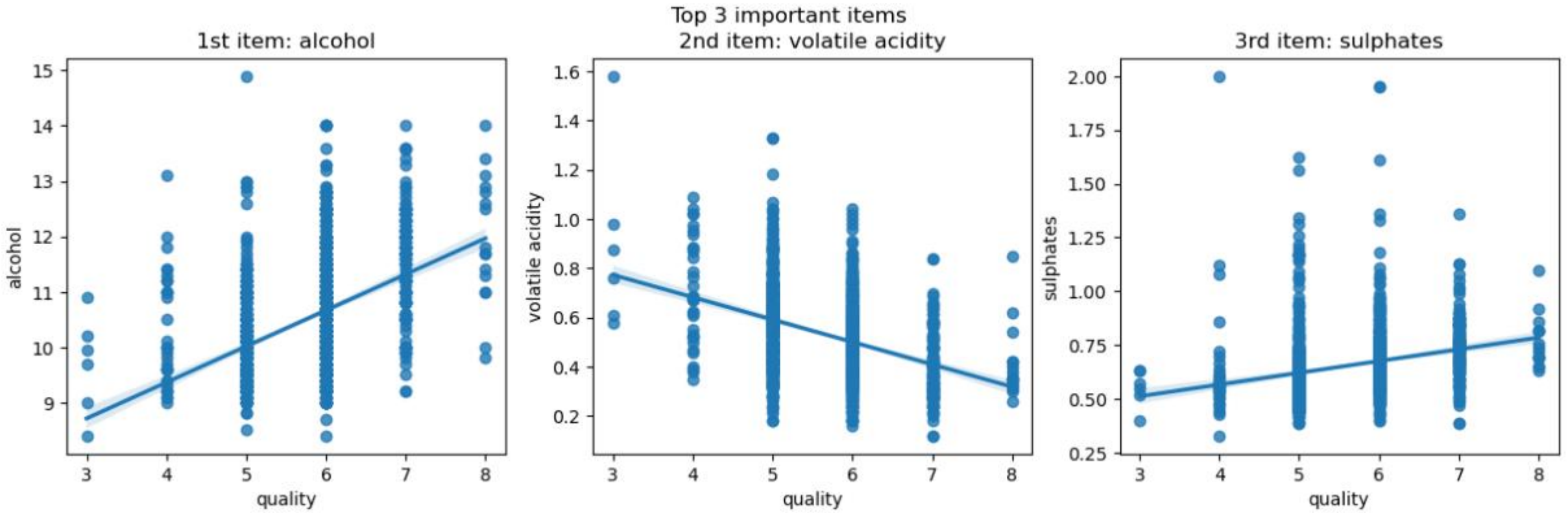
```
[0.         0.         0.01027408 0.01110531 0.02675301 0.02807349
 0.02904402 0.0739274  0.2345267  0.26734148 0.31895451]
Index(['residual sugar', 'pH', 'chlorides', 'free sulfur dioxide',
       'fixed acidity', 'density', 'total sulfur dioxide', 'citric acid',
       'sulphates', 'volatile acidity', 'alcohol'],
      dtype='object')
```



alcohol (0.32), volatile acidity (0.27), and sulphates (0.23) are the top 3 important items

3. Analysis the top 3 important items (2/2)

```
# Plot scatter chart and linear regression model fit
fig, axes = plt.subplots(1, 3, figsize=(15, 4))
fig.suptitle('Top 3 important items')
sns.regplot(ax=axes[0], data=data, x='quality', y='alcohol')
axes[0].set_title('1st item: alcohol')
sns.regplot(ax=axes[1], data=data, x='quality', y='volatile acidity')
axes[1].set_title('2nd item: volatile acidity')
sns.regplot(ax=axes[2], data=data, x='quality', y='sulphates')
axes[2].set_title('3rd item: sulphates')
```



4. Conclusion

According to our statistical results, good red wine is likely to have higher alcohol content, lower volatile acidity and higher sulfate content.