HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 2

BÁO CÁO CUỐI KỲ

Môn học: Nhập môn khoa học dữ liệu

Giảng viên:	***Tân Hạnh***
Sinh viên thực hiện:	***Trần Ngọc Khánh Văn***
Mã sinh viên:	***N20DCCN084***
Lớp:	***D20CQCNHT01-N***

Hồ Chí Minh, 2024

LỜI MỞ ĐẦU

Em xin gửi lời cảm ơn sâu sắc và chân thành tới thầy Tân Hạnh vì đã có những buổi dạy rất hay và giúp em học được thêm nhiều kiến thức có ích. Trong thời gian học với thầy em đã nhận được sự dạy dỗ, hướng dẫn nhiệt tình của thầy làm cho em có cái nhìn khách quan hơn cũng như là hiểu rõ hơn về khoa học dữ liệu ngành mà em đang muốn theo đuổi trong hiện tại. Em xin chân thành cảm ơn!

Em xin gửi lời cảm ơn cảm ơn ban lãnh đạo Học Viện Công nghệ Bưu Chính Viễn Thông cơ sở tại thanh phố Hồ Chí Minh đã dành thời gian quý báu của mình để trả lời các phiếu trắc nghiệm, tìm kiếm và cung cấp tư liệu, tư vấn, giúp đỡ chúng tôi hoàn thành bài tập cuối kỳ này.

Hồ Chí Minh, ngày 12 tháng 01 năm 2024

MỤC LỤC

Phần I: Nội dung thực hiện 4
1. Phương pháp hồi quy tuyến tính đa biến 4
2. Phương pháp hồi quy logistics 4
3. Cách chia tập dữ liệu 5
3.1. Tại sao nên chia tập dữ liệu 5
3.2. Cách chia tập dữ liệu 5
Phần II: Chương trình Python thực hiện việc huấn luyện và kiểm thử 7
1. Các mô-đun dùng trong bài 7
1.1 Pandas 7
1.2 Seaborn và Matplotlib Pyplot 7
1.3 Sklearn (Scikit-Learn) 7
2. Đọc dữ liệu và tiền xử lý dữ liệu 8
3. Chia dữ liệu thành tập huấn luyện và tập kiểm thử 15
4. Huấn luyện mô hình, kiểm thử và đánh giá dể mô tả độ chính xác của các phương pháp 15
4.1. Phương pháp hồi quy tuyến tính đa biến 16
4.2. Phương pháp hồi quy logistics 18
5. Cải thiện mô hình và bảng đánh giá 20
5.1. Cải thiện mô hình 20
5.2. Bảng đánh giá 22

Phần I: Nội dung thực hiện

1. Phương pháp hồi quy tuyến tính đa biến

Phương pháp hồi quy tuyến tính đa biến hay còn được gọi là Multiple Regression Linear là một kỹ thuật phân tích dữ liệu dự đoán giá trị của dữ liệu không xác định bằng cách sử dụng một giá trị dữ liệu liên quan được phát triển từ phưởng pháp hồi quy tuyến tính (Linear Regression).

Trên thực tế phương pháp hồi quy tuyến tính đa biến được dùng trong máy học để phân tích các tập dữ liệu lớn. Các nhà khoa học dữ liệu đầu tiên sẽ đào tạo thuật toán trên các tập dữ liệu đã biết hoặc được dán nhãn và sau đó sử dụng thuật toán để dự đoán các giá trị chưa biết.

Trong phân tích hồi quy tuyến tính đa biến, tập dữ liệu chứa một biến phụ thuộc và nhiều biến độc lập. Hàm đường hồi quy tuyến tính thay đổi để bao gồm nhiều yếu tố như sau:

$$Y= \beta0*X0 + \beta1X1 + \beta2X2+\dots\dots \beta nXn+ \varepsilon$$

Với:

Y là biến phụ thuộc

X0, X1, X2,….., Xn là các biến độc lập

β0, β1, β2,….., βn là các tham số

ε là các biến ngẫu nhiên gọi là số hạng sai số

2. Phương pháp hồi quy logistics

Phương pháp hồi quy logistics hay còn được biết đến là Logistic Regression là một kỹ thuật phân tích dữ liệu sử dụng toán học để tìm ra mối quan hệ giữa hai yếu tố dữ liệu. Sau đó, kỹ thuật này sử dụng mối quan hệ đã tìm được để dự đoán giá trị của những yếu tố đó dựa trên yếu tố còn lại. Dự đoán thường cho ra một số kết quả hữu hạn, như có hoặc không.

Hồi quy logistic là một kỹ thuật quan trọng trong lĩnh vực trí tuệ nhân tạo và máy học (AI/ML). Mô hình ML là các chương trình phần mềm có thể được đào tạo để thực hiện các tác vụ xử lý dữ liệu phức tạp mà không cần sự can thiệp của con người. Mô hình ML được xây dựng bằng hồi quy logistic có thể giúp các tổ chức thu được thông tin chuyên sâu hữu ích từ dữ liệu kinh doanh của mình. Họ có thể sử dụng những thông tin

chuyên sâu này để phân tích dự đoán nhằm giảm chi phí hoạt động, tăng độ hiệu quả và đổi chỉnh quy mô nhanh hơn. Ví dụ: doanh nghiệp có thể khám phá các mẫu hình cải thiện khả năng giữ chân nhân viên hoặc tạo ra thiết kế sản phẩm mang về nhiều lợi nhuận hơn.

Một giải pháp để phân loại là hồi quy logistic. Thay vì điều chỉnh một đường thẳng hoặc siêu phẳng, mô hình hồi quy logistic sử dụng hàm logistic để ép đầu ra của một phương trình tuyến tính giữa 0 và 1. Hàm logistic được định nghĩa là:

$$\text{Logistic}()x = \frac{1}{1+exp(n)}$$

3. Cách chia tập dữ liệu

3.1. Tại sao nên chia tập dữ liệu

Trước khi tìm hiểu về cách chia tập dữ liệu thì ta cần nói đến hai vấn đề quan trọng trong máy học đó là overfitting và underfitting. Với các mô hình underfitting cho kết quả tệ trên cả dữ liệu quan sát sử dụng để huấn luyện lẫn dữ liệu chưa được quan sát trong thực tế, còn overfitting cho kết quả rất (quá) tốt trên dữ hiệu huấn luyện nhưng lại quá kém khi sử dụng trong thực tế.

Trong machine learning, chúng ta cần kiểm thử để dự đoán khả năng hoạt động hiệu quả của mô hình trên thực tế. Có rất nhiều cách để kiểm thử và đánh giá hiệu năng của mô hình. Cách đơn giản nhất là dùng thử nhưng điều này mang lại rủi ro và tốn kém quá cao. Vì vậy cách chia tập dữ liệu ra là cách hiệu quả nhất.

3.2. Cách chia tập dữ liệu

Tập huấn luyện là tập dữ lệu dùng cho việc huấn luyện mô hình. Các thuật toán máy học sẽ học các mô hình từ tập này.

Tập kiểm thử chính là tập dữ liệu dùng cho việc xác định độ chính xác hoặc sai số của mô hình máy đã huấn luyện từ tập dữ liệu huấn luyện. Chúng ta biết nhãn thực của mọi điểm trong tập hợp dữ liệu kiểm thử này, nhưng chúng ta sẽ tạm thời giả vờ như không biết và đưa các giá trị đầu vào của tập vào mô hình dự đoán để nhận kết quả dự đoán đầu ra. Sau đó chúng ta có thể nhìn vào các nhãn thực và so sánh nó với kết quả dự đoán của các đầu vào tương ứng này và xem liệu mô hình có dự đoán đúng hay không. Việc tính tổng trung bình của toàn bộ các lỗi này chúng ta có thể tính toán được lỗi dự

đoán trên tập kiểm thử.

Có 3 cách chia dữ liệu thường được sử dụng đó là:

- Hold-out

-Train-Validation-Test Split

- Cross Validation

Ở bài tập này mục đích của chúng ta là kiểm chứng mô hình được huấn luyện vậy nên phương pháp chia **hold-out** chính là phương pháp phù hợp nhất. Theo phương pháp hold-out chúng ta sẽ dùng 80% tập dữ liệu để huấn luyện và 20% còn lại dùng để kiểm thử vì đây là cách chia hợp lý nhất. Nếu tập kiểm thử quá nhỏ thì việc kiểm thử có thể không đáng tin cậy và có độ bất ổn cao. Dựa trên mức độ "may mắn" khi chia dữ liệu mà các độ đo này có thể nhảy lên cao hoặc rất thấp

Phần II: Chương trình Python thực hiện việc huấn luyện và kiểm thử

1. Các mô-đun dùng trong bài

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import style

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, ConfusionMatrixDisplay
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

1.1 Pandas

Pandas là một thư viện phổ biến được sử dụng rộng rãi trong việc phân tích và xử lý dữ liệu trong Python. Nó cung cấp các cấu trúc dữ liệu và công cụ mạnh mẽ để làm việc với dữ liệu số hóa, chẳng hạn như bảng dữ liệu (DataFrame), và cung cấp hỗ trợ cho các thao tác phân tích dữ liệu như lọc, nhóm, sắp xếp, tính toán và trực quan hóa.

1.2 Seaborn và Matplotlib Pyplot

Seaborn là một thư viện trực quan hóa dữ liệu mạnh mẽ cho Python. Nó được xây dựng dựa trên thư viện Matplotlib và cung cấp các chức năng cao cấp để tạo ra các biểu đồ hấp dẫn và chuyên nghiệp. Seaborn cung cấp một loạt các chủ đề trước định nghĩa để tùy chỉnh giao diện trực quan, giúp tạo ra các biểu đồ có màu sắc và kiểu dáng hấp dẫn. Nó cũng hỗ trợ việc thực hiện các biểu đồ phân tán, biểu đồ đường, biểu đồ hộp, heatmap, biểu đồ violin và nhiều loại biểu đồ khác.

Matplotlib.pyplot là một module trong thư viện Matplotlib của Python, và nó được sử dụng để tạo và hiển thị các biểu đồ và hình vẽ trong môi trường tương tác.

1.3 Sklearn (Scikit-Learn)

Scikit-learn (hay sklearn) là một thư viện phổ biến trong Python được sử dụng cho machine learning và data mining. Nó cung cấp các công cụ và thuật toán tiêu chuẩn để xây dựng và áp dụng các mô hình học máy trên dữ liệu.

Scikit-learn hỗ trợ nhiều loại tác vụ machine learning như phân loại, hồi quy, gom nhóm, trích xuất đặc trưng, giảm chiều dữ liệu và phân tích mạng phức tạp. Nó cũng cung cấp các công cụ để đánh giá và tinh chỉnh mô hình, bao gồm chia tập dữ liệu, cross-validation, và các phương pháp đánh giá hiệu suất.

2. Đọc dữ liệu và tiền xử lý dữ liệu

```
#Read data from csv
#Change the file name to switch between red wine and white wine
redWine_data = pd.read_csv('winequality-red.csv',sep=';')
whiteWine_data = pd.read_csv('winequality-white.csv',sep=';')
```

Đọc dữ liệu từ file .csv và chuyển vào dạng data frame của thư viện python.

```
#Shape of the data
whiteWine_data.shape
```

```
(4898, 12)
```

```
redWine_data.shape
```

```
(1599, 12)
```

Xem kích thước dữ liệu của data set red win và white wine.

```
redWine_data.isnull().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
dtype: int64
```

```
whiteWine_data.isnull().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
dtype: int64
```

Kiểm tra xem dữ liệu có bị lỗi không, việc làm việc với dữ liệu lỗi khi lập trình sẽ có rủi ro rất cao và có thể dẫn đến kết quả tệ.

```
redWine_data['quality'].value_counts()
```

```
quality
5    681
6    638
7    199
4     53
8     18
3     10
Name: count, dtype: int64
```

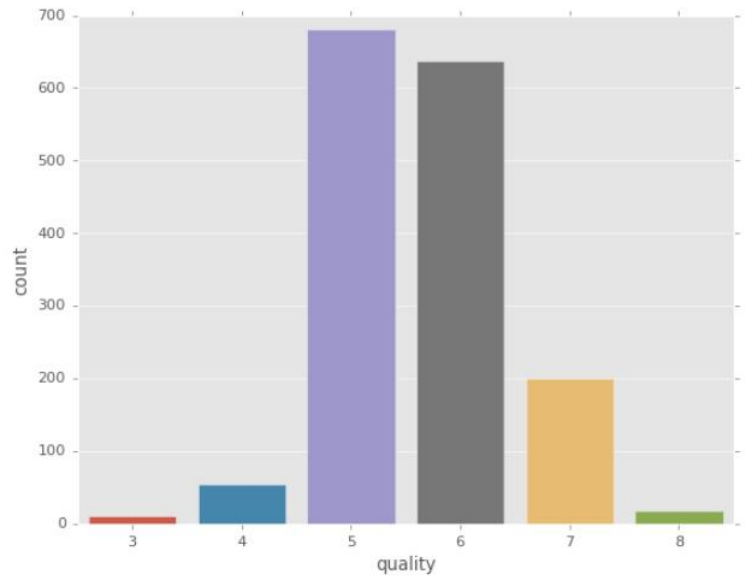
```
whiteWine_data['quality'].value_counts()
```

```
quality
6    2198
5    1457
7     880
8     175
4     163
3      20
9       5
Name: count, dtype: int64
```

Xem thử các loại chất lượng rượu khác nhau của 2 data set ta thấy chất lượng rượu của white wine có thể có chất lượng bằng 9 còn red win thì không việc này có thể dẫn đến độ chính xác của mô hình dự đoán cho white wine có thể giảm vì số lượng biến của white nhiều hơn.

```
style.use('ggplot')
sns.countplot(x='quality' ,data=redWine_data)
```

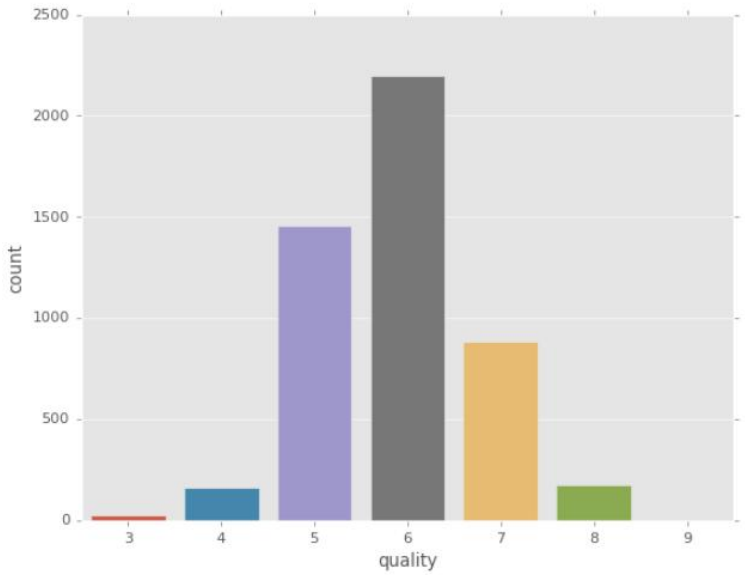
```
<Axes: xlabel='quality', ylabel='count'>
```



Biểu đồ thể hiện sự phân bố của chất lượng rượu trong data set red wine.

```
style.use('ggplot')
sns.countplot(x='quality' ,data=whiteWine_data)
```

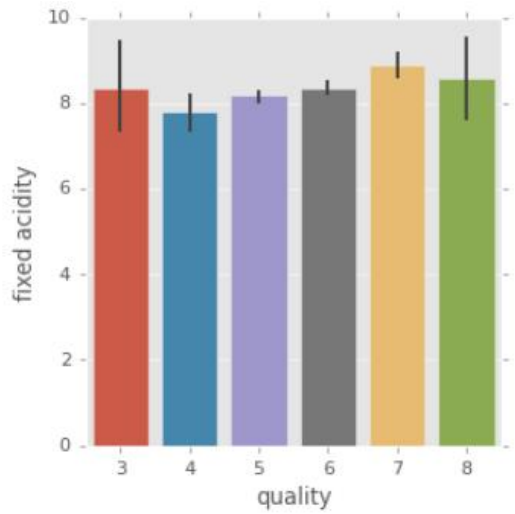
```
<Axes: xlabel='quality', ylabel='count'>
```

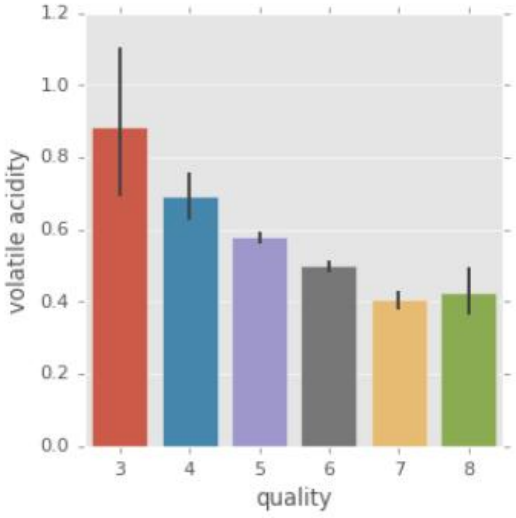


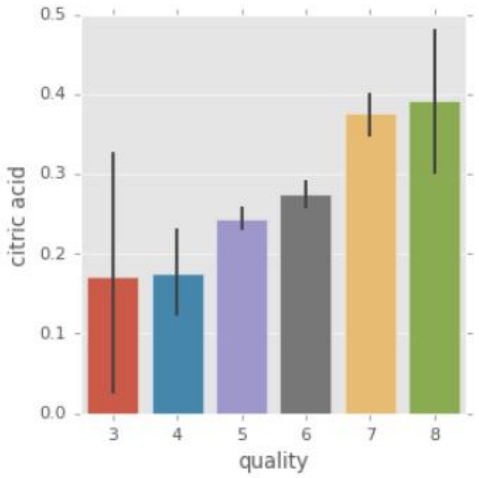
Biểu đồ thể hiện sự phân bố của chất lượng rượu trong data set white wine.

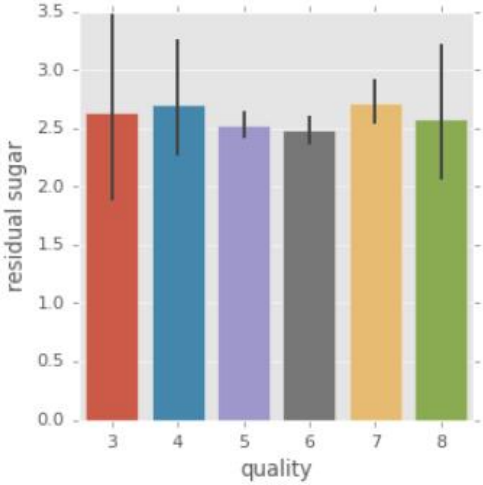
```
#check which element effect the quality the most
s=["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxide",
   "total sulfur dioxide","density","pH","sulphates","alcohol"]
for i in s:
    plot = plt.figure(figsize=(4,4))
    sns.barplot(x='quality', y=i, data=redWine_data)
```

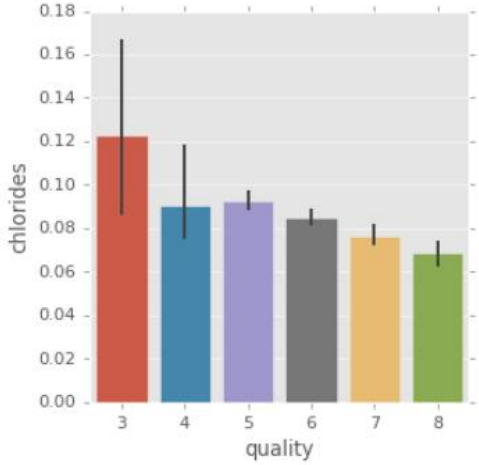
Đoạn code trên sẽ vẽ ra biểu đồ sự ảnh hưởng của từng yếu tố đến chất lượng cửa rượu trong data set red wine.

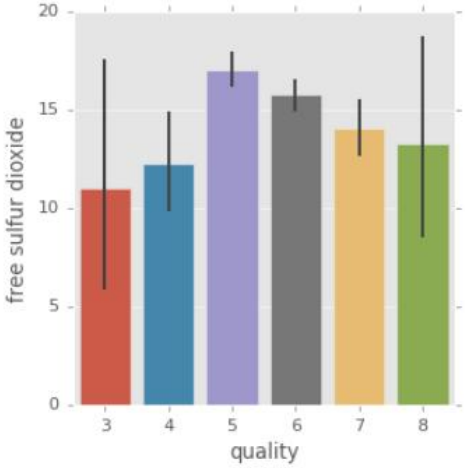


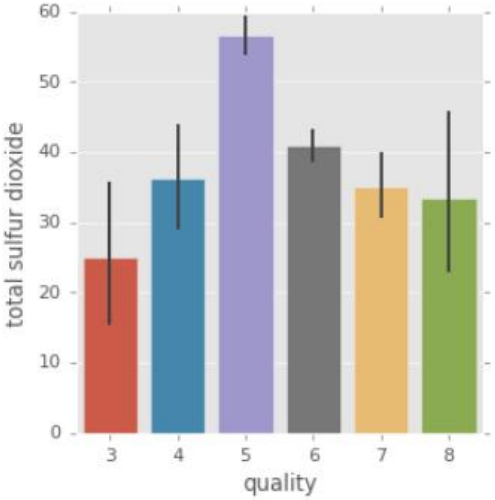


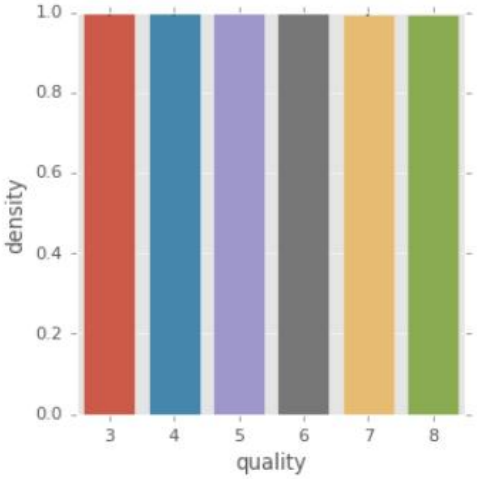

citric acid
0.0
0.1
0.2
0.3
0.4
0.5
3
4
5
6
7
8
quality

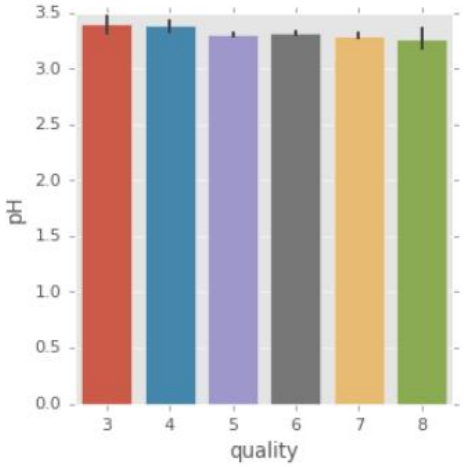

residual sugar
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
3
4
5
6
7
8
quality

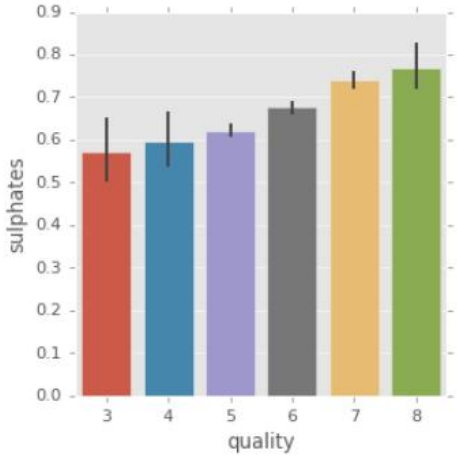

chlorides
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
3
4
5
6
7
8
quality

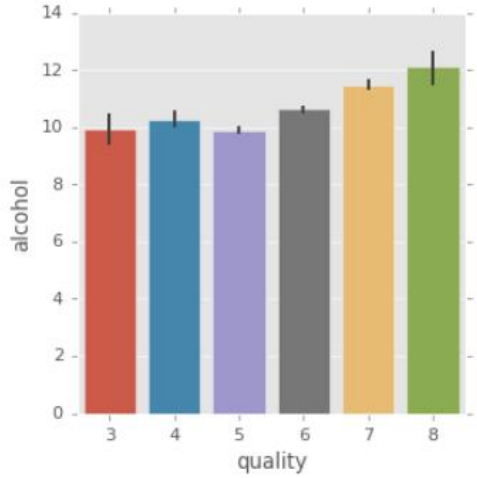

free sulfur dioxide
0
5
10
15
20
3
4
5
6
7
8
quality


total sulfur dioxide
0
10
20
30
40
50
60
3
4
5
6
7
8
quality


density
0.0
0.2
0.4
0.6
0.8
1.0
3
4
5
6
7
8
quality

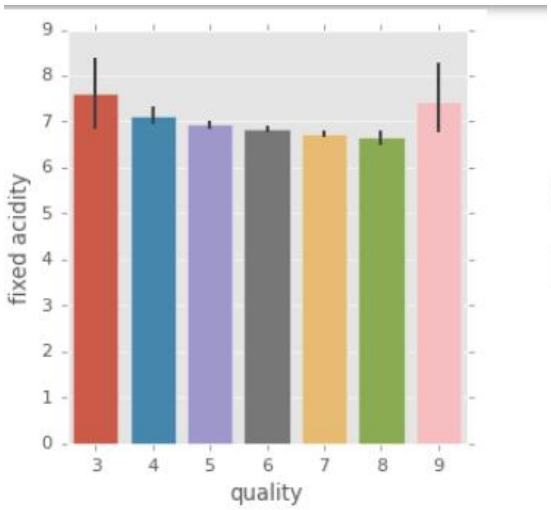


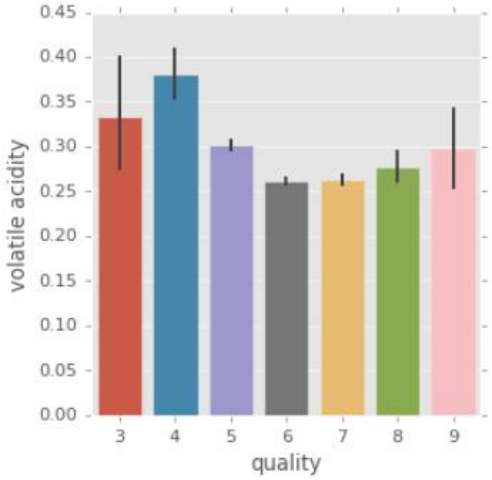


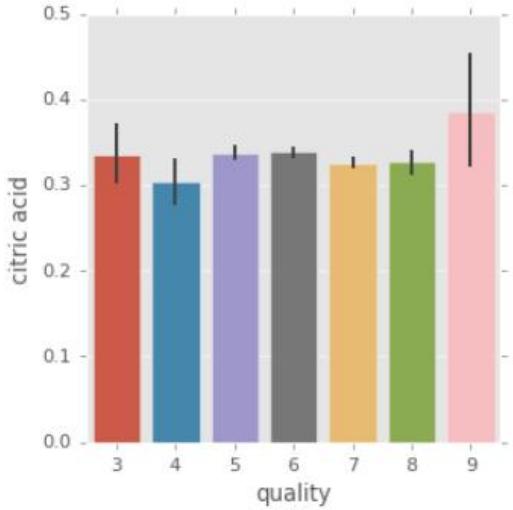


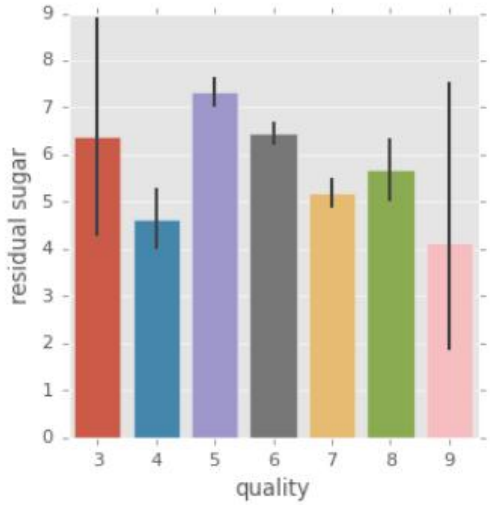

```
#check which element effect the quality the most
s=["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxide",
   "total sulfur dioxide","density","pH","sulphates","alcohol"]
for i in s:
    plot = plt.figure(figsize=(4,4))
    sns.barplot(x='quality', y=i, data=whiteWine_data)
```

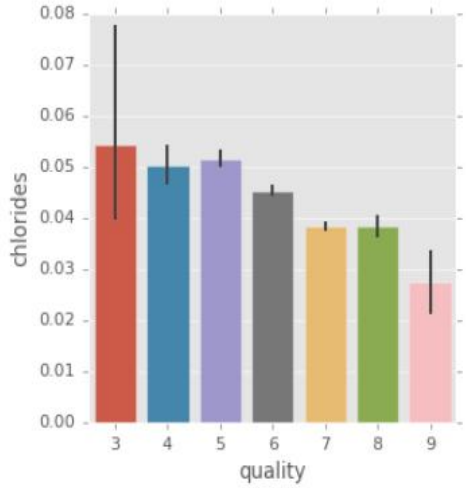
Đoạn code trên sẽ vẽ ra biểu đồ sự ảnh hưởng của từng yếu tố đến chất lượng cửa rượu trong data set white wine.

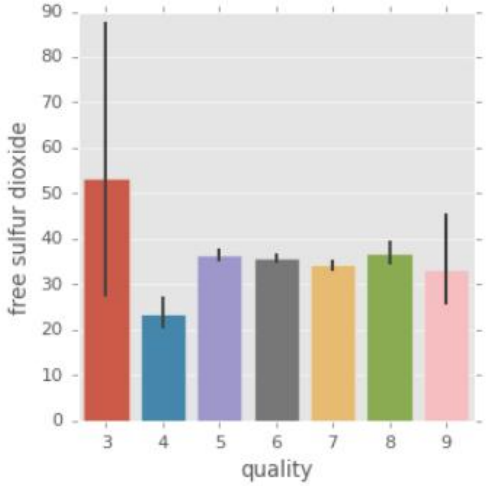

fixed acidity
quality

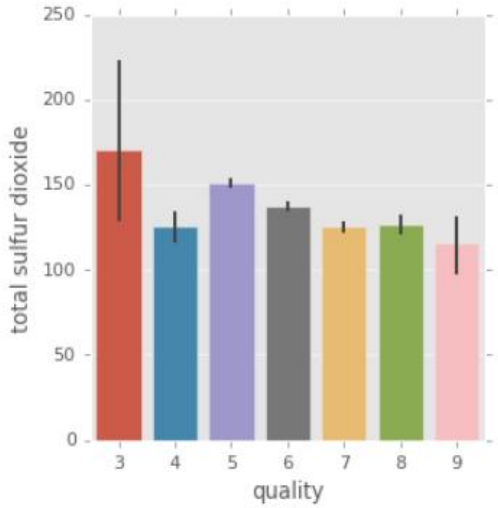

volatile acidity
quality

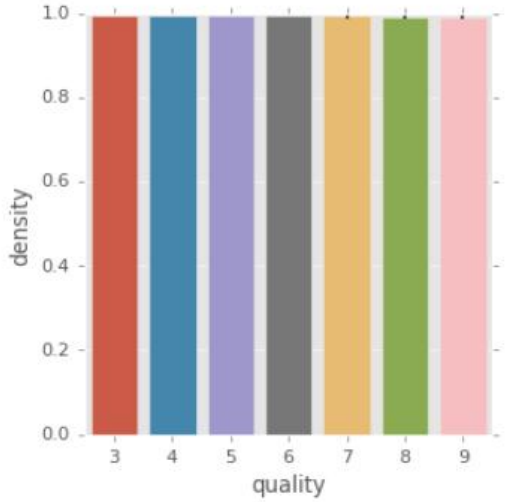

citric acid
quality

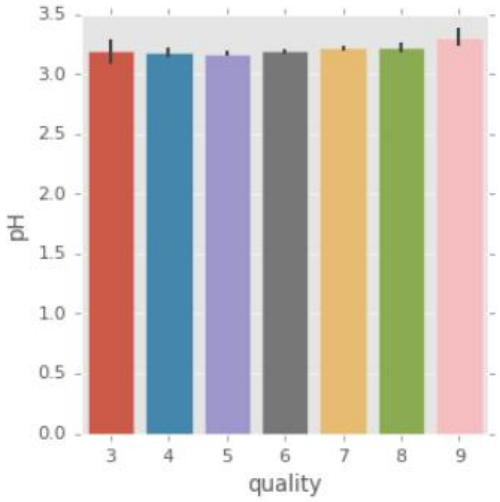

residual sugar
quality

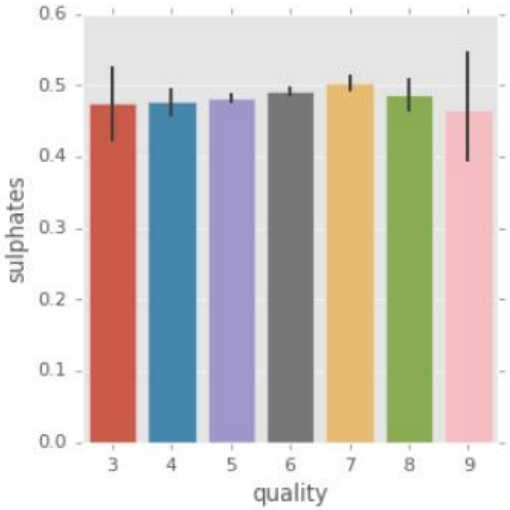

chlorides
quality

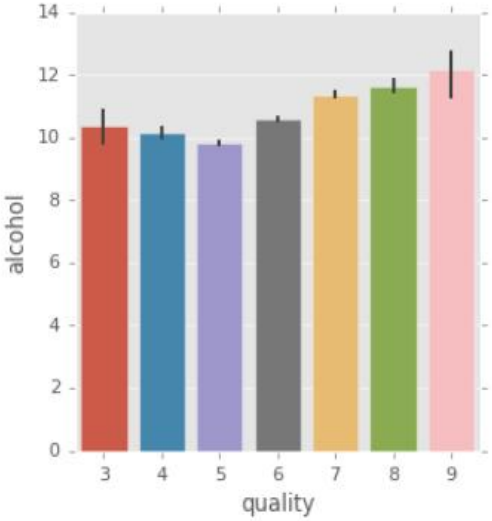

free sulfur dioxide
quality


total sulfur dioxide
250
200
150
100
50
0
3
4
5
6
7
8
9
quality


density
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
9
quality


pH
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
3
4
5
6
7
8
9
quality


sulphates
0.6
0.5
0.4
0.3
0.2
0.1
0.0
3
4
5
6
7
8
9
quality


alcohol
14
12
10
8
6
4
2
0
3
4
5
6
7
8
9
quality

3. Chia dữ liệu thành tập huấn luyện và tập kiểm thử

```
X = redWine_data.drop('quality', axis=1)
y = redWine_data['quality']
X1 = whiteWine_data.drop('quality', axis=1)
y1 = whiteWine_data['quality']
```

Ta chia dữ liệu thành x và y. X là các biến ảnh hưởng đến chất lượng của rượu, y là chất lượng của rượu

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=42)
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y1, test_size=0.2, random_state=42)
```

Tiến hành phân chia dữ liệu thành 2 tập, tập huấn luyện và tập kiểm thử với tỷ lệ là:

- Tập huấn luyện 80%
- Tập kiểm thử 20%

Test_size chính là tập dữ liệu và 0.2 chính là 20%

```
print("X_train ", X_train.shape)
print("y_train ", y_train.shape)
print("X_test ", X_test.shape)
print("y_test ", y_test.shape)
```

```
X_train  (1279, 11)
y_train  (1279,)
X_test  (320, 11)
y_test  (320,)
```

```
print("X1_train ", X1_train.shape)
print("y1_train ", y1_train.shape)
print("X1_test ", X1_test.shape)
print("y1_test ", y1_test.shape)
```

```
X1_train  (3918, 11)
y1_train  (3918,)
X1_test  (980, 11)
y1_test  (980,)
```

Đoạn code trên giúp ta thấy được số dữ liệu của tập kiểm thử và tập huấn luyện trên cả hai data set.

4. Huấn luyện mô hình, kiểm thử và đánh giá để mô tả độ chính xác của các phương pháp

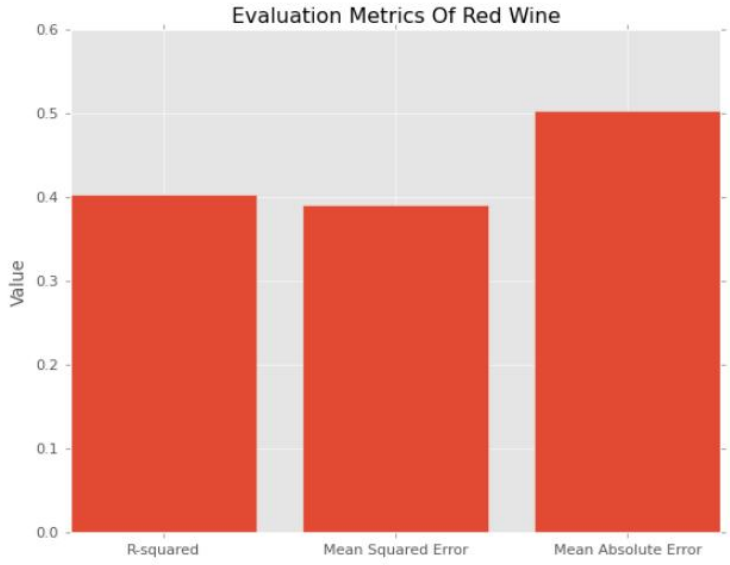
4.1. Phương pháp hồi quy tuyến tính đa biến

4.1.1. Rượu vang đỏ

```
# Create logistic regression
linear_reg = LinearRegression()

# Training variable
linear_reg.fit(X_train, y_train)

# Prediction
linear_reg_pred = linear_reg.predict(X_test)

#Mean squared error
mse = mean_squared_error(y_test, linear_reg_pred)
print("Mean Squared Error Of Red Wine: {:.2f}".format(mse))

# Calculate R-squared
r2 = r2_score(y_test, linear_reg_pred)
print("R-Squared Of Red Wine: {:.2f}".format(r2))

# Calculate mean absolute error
mae = mean_absolute_error(y_test, linear_reg_pred)
print("Mean Absolute Error Of Red Wine: {:.2f}".format(mae))
```

```
Mean Squared Error Of Red Wine: 0.39
R-Squared Of Red Wine: 0.40
Mean Absolute Error Of Red Wine: 0.50
```

```
metrics = ['R-squared', 'Mean Squared Error', 'Mean Absolute Error']
values = [r2, mse, mae]

plt.bar(metrics, values)
plt.ylabel('Value')
plt.title('Evaluation Metrics Of Red Wine')
plt.show()
```



Phía trên cho ta thấy những chỉ số R-squared, Mean Squared Error, Mean Absolute Error là những thông số chỉ của mô hình sử dụng phương pháp hồi quy tuyến tính đa biến cho data set red wine có ý nghĩa:

- Mean Squared Error (MSE): MSE đo lường sai số trung bình của dự đoán so với giá trị thực tế, được tính bằng cách lấy trung bình của bình phương sai số. Giá trị MSE càng thấp thì mô hình càng tốt, với giá trị 0 là tốt nhất.Trong trường hợp này, MSE là 0.39, cho thấy sai số trung bình bình phương giữa dự đoán và giá trị thực tế là 0.39.

- R-Squared (R^2): R-squared là một độ đo thống kê biểu thị tỷ lệ phần trăm phương sai của biến phụ thuộc (biến mục tiêu) có thể được giải thích bởi các biến độc lập (các thuộc tính) trong mô hình. R-squared có giá trị từ 0 đến 1, với giá trị cao hơn cho thấy mô hình phù hợp tốt hơn. Giá trị R^2 là 0.40 cho thấy khoảng 40% phương sai của biến mục tiêu được giải thích bởi các biến độc lập trong mô hình của bạn.

- Mean Absolute Error (MAE): MAE đo lường sai số trung bình tuyệt đối giữa dự đoán và giá trị thực tế. Nó cung cấp một phép đo sai số dễ hiểu hơn so với MSE vì nó có cùng đơn vị với biến mục tiêu. Tương tự như MSE, giá trị MAE càng thấp thì mô hình càng tốt, với giá trị 0 là tốt nhất. Trong trường hợp này, MAE là 0.50, cho thấy sai số trung bình tuyệt đối giữa dự đoán và giá trị thực tế là 0.50.

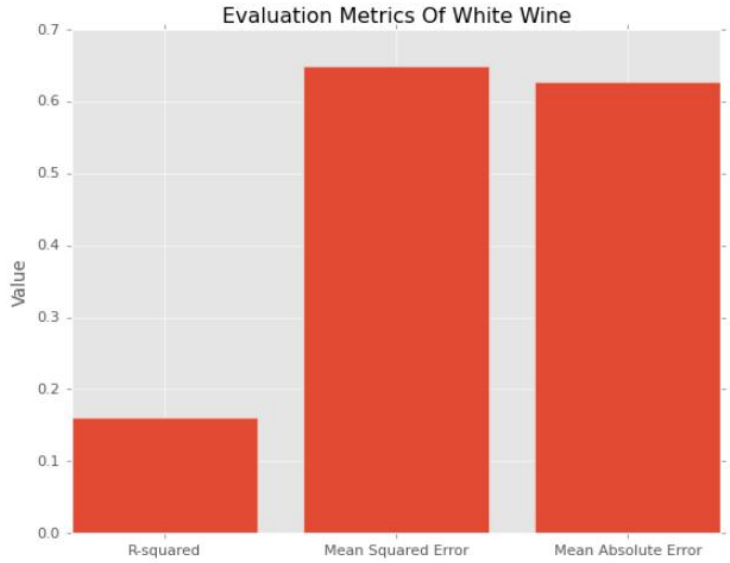
Từ đó ta có kết luận rằng: Mô hình có R-squared thấp hơn 50% thường được coi là một hiệu suất yếu và mô hình không thể giải thích tốt sự biến thiên của biến mục tiêu một mô hình tốt thường có R^2 dao động từ 70% đến 90% , mô hình có MSE và MAE thấp cũng chỉ ra được hiệu xuất dự đoán chất lượng của rượu trong mô hình là kém.

4.1.2. Rượu vang trắng

```
# Create logistic regression
linear_reg1 = LinearRegression()

# Training variable
linear_reg1.fit(X1_train, y1_train)

# Prediction
linear_reg1_pred = linear_reg.predict(X1_test)

#Mean squared error
mse1 = mean_squared_error(y1_test, linear_reg1_pred)
print("Mean Squared Error Of White Wine: {:.2f}".format(mse1))

# Calculate R-squared
r21 = r2_score(y1_test, linear_reg1_pred)
print("R-Squared Of White Wine: {:.2f}".format(r21))

# Calculate mean absolute error
mae1 = mean_absolute_error(y1_test, linear_reg1_pred)
print("Mean Absolute Error Of White Wine: {:.2f}".format(mae1))
```

```
Mean Squared Error Of White Wine: 0.65
R-Squared Of White Wine: 0.16
Mean Absolute Error Of White Wine: 0.63
```

```
metrics = ['R-squared', 'Mean Squared Error', 'Mean Absolute Error']
values = [r21, mse1, mae1]

plt.bar(metrics, values)
plt.ylabel('Value')
plt.title('Evaluation Metrics Of White Wine')
plt.show()
```



Với mô hình huấn luyện dựa data set white wine do sự biến thiên của chất lượng rượu lớn hơn so với data set red wine dẫn đến độ chính xác của mô hình giảm mặc dù có lượng dữ liệu lớn hơn nhưng mô hình này vẫn tỏ ra thua kém so với mô hình cho data set red wine.

4.2. Phương pháp hồi quy logistics

4.2.1. Rượu vang đỏ

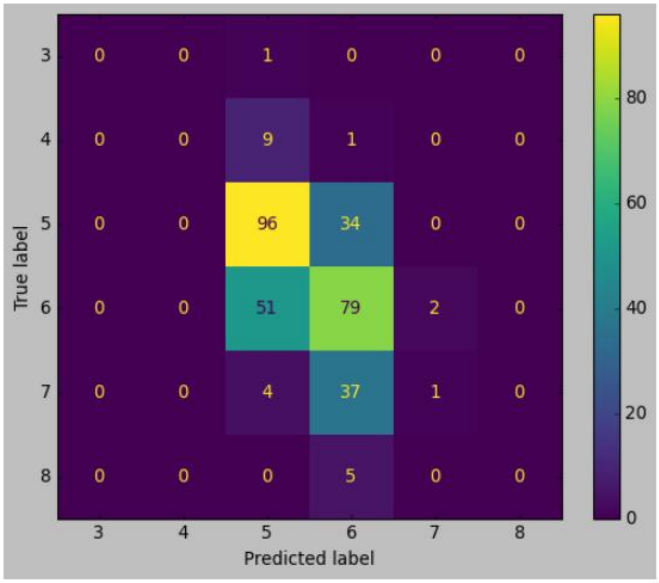
```
# Create logistic regression
log_reg= LogisticRegression()

# Training variable
log_reg.fit(X_train, y_train)

# Prediction
log_reg_pred = log_reg.predict(X_test)

# Accurance testing
log_reg_acc = accuracy_score(log_reg_pred, y_test)
print("Test accuracy is: {:.2f}%".format(log_reg_acc*100))
```

```
Test accuracy is: 55.00%
```

Số liệu chỉ ta thấy độ chính xác khi kiểm thử là không cao chỉ khoảng 55% chứng tỏ mô hình không hoạt động tốt.

```
#Confusion matrix
style.use('classic')
cm = confusion_matrix(y_test, log_reg_pred, labels=log_reg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix= cm, display_labels=log_reg.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2272fbc9ed0>
```



Biểu đồ dựa trên kết quả dự đoán của mô hình so với kết quả thực tế.

4.2.2. Rượu vang trắng

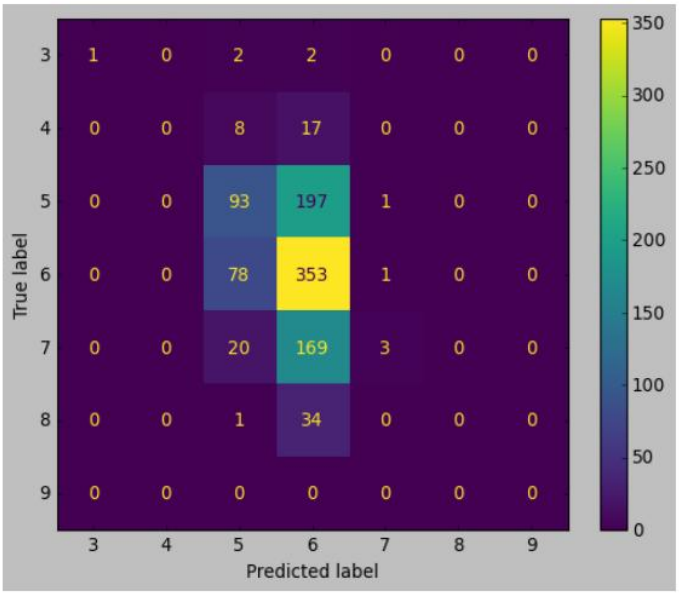
```
# Create logistic regression
log_reg1= LogisticRegression()

# Training variable
log_reg1.fit(X1_train, y1_train)

# Prediction
log_reg1_pred = log_reg1.predict(X_test)

# Accurance testing
log_reg1_acc = accuracy_score(log_reg1_pred, y_test)
print("Test accuracy is: {:.2f}%".format(log_reg1_acc*100))
```

```
Test accuracy is: 45.62%
```

Cũng giống với phương pháp hồi quy tuyến tính đa biến thì mô hình được huấn luyện của data set white wine vẫn chô độ chính xác thấp hơn mô hình của red wine.

```
#Confusion matrix
#TN (True Negative), FN (False Negative), TP (True Positive) and FP (False Positive)
style.use('classic')
cm1 = confusion_matrix(y1_test, log_reg1_pred, labels=log_reg1.classes_)
disp1 = ConfusionMatrixDisplay(confusion_matrix= cm1, display_labels=log_reg1.classes_)
disp1.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x22726f8f690>
```



Biểu đồ dựa trên kết quả dự đoán của mô hình so với kết quả thực tế. Mặc dù số lần dự đoán đúng là nhiều hơn mô hình với data set red wine thế nhưng data set white wine vẫn có nhiều dữ liệu hơn rất nhiều nên tỷ lệ dự đoán đúng cũng thấp hơn.

5. Cải thiện mô hình và bảng đánh giá

5.1. Cải thiện mô hình

```
# #Quality >7 mean good wine and = 1
# #Quality <7 mean not good wine and =0
redWine_data['quality'] = redWine_data.quality.apply(lambda x:1 if x>=7 else 0)
whiteWine_data['quality'] = whiteWine_data.quality.apply(lambda x:1 if x>=7 else 0)
```

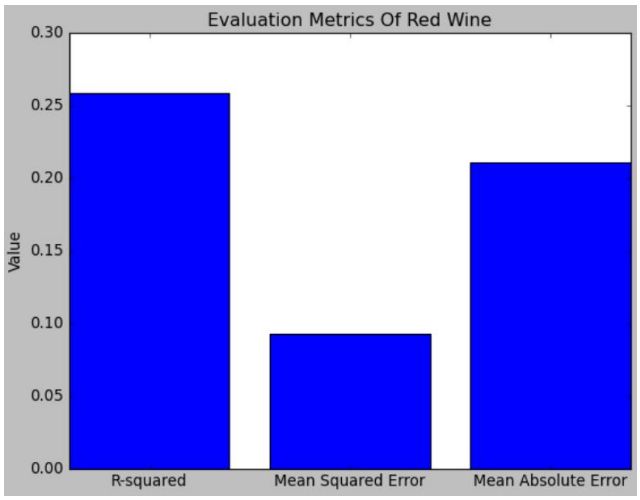
Phương pháp cải thiện em sử dụng ở đây chính là đơn giản hóa phân loại. Cụ thể chúng ta sẽ chuyển đổi chất lượng rượu vang thành một biến nhị phân giúp đơn giản hóa việc phân loại. Thay vì xử lý nhiều giá trị đánh giá chất lượng khác nhau, bạn chỉ cần quan tâm đến hai giá trị 0 và 1. Bằng cách chuyển đổi thành một biến nhị phân, bạn có thể tập trung vào việc phân loại rượu vang thành hai nhóm: rượu vang tốt và rượu vang không tốt. Điều này có thể hữu ích trong việc phân tích và so sánh các đặc điểm và yếu tố ảnh hưởng đến chất lượng rượu vang.

Đoạn code trên sẽ chuyển đổi các đánh giá chất lượng của rượu vang thành một phân loại nhị phân: rượu vang có đánh giá chất lượng từ 7 trở lên được gán nhãn là 1 (chất

lượng tốt), trong khi rượu vang có đánh giá chất lượng dưới 7 được gán nhãn là 0 (chất lượng không tốt). Chúng ta sẽ đưa đoạn code này vào trước khi chia các tập huấn luyện và kiểm thử.

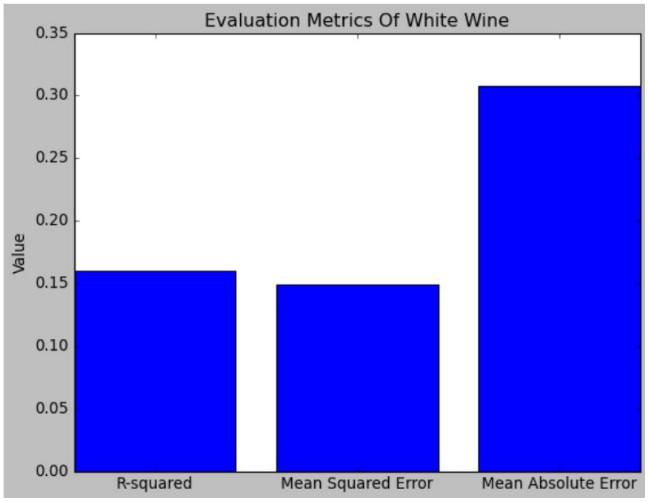
Red Wine

```
Mean Squared Error Of Red Wine: 0.09
R-Squared Of Red Wine: 0.26
Mean Absolute Error Of Red Wine: 0.21
```



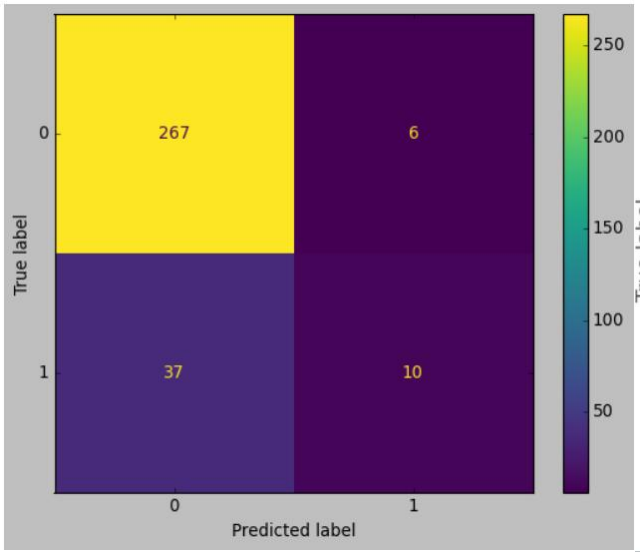
White Wine

```
Mean Squared Error Of White Wine: 0.15
R-Squared Of White Wine: 0.16
Mean Absolute Error Of White Wine: 0.31
```



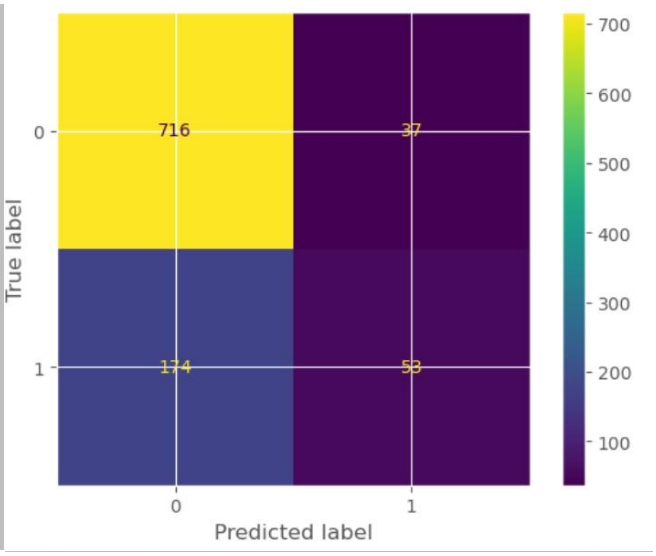
Red Wine

```
Test accuracy of red wine is: 86.56%
```



White Wine

```
Test accuracy of white wine is: 86.25%
```



Ta thấy sau khi cải thiện mô hình thì cả 2 phương pháp đều đem lại hiệu quả tốt hơn rất nhiều tuy nhiên R-squred của phương pháp hồi quy tuyến tính đa biến lại giảm cho

thấy mô hình giảm khả năng giải thích sự biến thiên của mục tiêu vậy ta có thể suy ra đôi khi việc giảm số biến kết quả có thể có tác dụng tích cực hơn ở phương pháp nhưng cũng có thể đem lại hiệu quả tiêu cực tới một vài khả năng của mô hình. Quyết định chuyển đổi các biến kết quả thành một biến nhị phân hay giữ nguyên giá trị đánh giá là tùy thuộc vào mục tiêu và yêu cầu cụ thể của bạn trong quá trình xử lý và phân tích dữ liệu.

5.2. Bảng đánh giá

	Kết quả ban đầu	
	Red Wine	*White Wine*
Mutiple Linear Regression	*MSE: 0.39,* R^2 *: 0.40, MAE : 0.50* *Mô hình có tỷ lệ dự đoán đúng thấp do cả chỉ số MSE và MAE đều khá cao.*	*MSE: 0.65,* R^2*: 0.16, MAE : 0.63* *Cũng có tỷ lệ dự đoán đúng thấp, tỷ lệ dự đoán đúng thấp hơn red wine.*
Logistics Regression	*Tỷ lệ dự đoán đúng chỉ có 55% thấp so với thực tế.*	*Tỷ lệ dự đoán đúng chỉ có 55% thấp so với thực tế và thấp hơn red wine.*

	Kết quả sau khi giảm biến kết quả	
	Red Wine	*White Wine*
Mutiple Linear Regression	*MSE: 0.09,* R^2 *: 0.26, MAE : 0.21* *Chỉ số MSE và MAE thấp hơn hẳn khi chưa tối ưu vậy nên tỷ dự đoán đúng cũng tăng mạnh tuy nhiên* R^2 cũng giảm nên mô hình giảm khả năng dự đoán sự biến thiên kết quả.	*MSE: 0.15,* R^2*: 0.16, MAE : 0.31* *MSE và MAE cũng giảm mạnh tuy nhiên vẫn không thể bằng red win, chỉ số* R^2 giữ nguyên.
Logistics Regression	*Tỷ lệ dự đoán đúng tăng lên 86.56% tăng cao thấy rõ suy ra mô hình hoạt động tốt hơn sau khi tối ưu.*	*Tỷ lệ dự đoán đúng tăng lên 86.25% và gần bằng với tỷ lệ dự đoán đúng của red wine*