Final Team Project

Andrew Kim, Luis Perez, Renetta Nelson

October 17, 2022

Problem Statement

> The purpose of this project is to automate the wine selection process in order to increase profit and build on the business's reputation. This will be done by implementing a model that predicts the quality of the wine. The profit margin of restaurants is approximately 70%. This means that over half the profit of these business types come from wine. On the other hand, there are also major expenses that pertain to wine as well. From vendors to sommeliers, there are dozens of additional expenses when it comes to finding and purchasing good quality wine. The profits of the business can no longer support the expenses of the wine selection process. Within a few months, the expenses will exceed the profits of the business and the business will have to close down. The automation of the wine selection process will reduce the expenses by approximately 25%, allowing the business to build its finances and stay in business.

In [129…

```
#Import Libraries

import pandas as pd
import numpy as np
import random
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, roc_curve
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from dmba import classificationSummary, gainsChart, liftChart
```

```
import scikitplot as skplt
import matplotlib.pyplot as plt
from dmba.metric import AIC_score
import statsmodels.api as sm
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from dmba import classificationSummary
from sklearn.metrics import r2_score, plot_confusion_matrix, multilabel_confusion_matrix, ConfusionMatrixDisplay
import seaborn as sns

warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model
import matplotlib.pylab as plt
import dmba
from dmba import regressionSummary
from dmba import adjusted_r2_score, AIC_score, BIC_score
from sklearn.cluster import KMeans
```

In [153…

```
#Load dataset and put into a data frame

redwine_data = pd.read_csv("winequality-red.csv")

redwine_df = pd.DataFrame(redwine_data)

#Display first five rows of dataframe to confirm

redwine_df.head()
```

Out[153…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Data Preprocessing

> Data Preprocessing Explanation for this section -> Can do a quick summary here, talking about each of the processes that are being done for data preprocessing

In [7]:
```
# Check type of variables
redwine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [8]:
```
redwine_df.describe()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	1(
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	1(
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	1'
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	1⁴

In [9]:
```
# Check for data size
redwine_df.shape
```

Out[9]:
```
(1599, 12)
```

In [10]:
```
redwine_df['quality'].unique()
```

Out[10]:
```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [11]:
```
redwine_df.isna().sum()
```

Out[11]:
```
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
```

In [12]:
```
redwine_df.dtypes
```

Out[12]:
```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

In [154…
```
# Removing Outliers

#based on the boxplots total sulfur dioxide has many outliers
d1= redwine_df['total sulfur dioxide']
mean = np.mean(redwine_df['total sulfur dioxide'])
std = np.std(redwine_df['total sulfur dioxide'])
print('mean of the dataset is', mean)
print('std. deviation is', std)


#z method
#total sulfur dioxide
out=[]
def Zscore_outlier(df):
    m = np.mean(df)
    sd = np.std(df)
    for i in df:
        z = (i-m)/sd
        if np.abs(z) > 3:
            out.append(i)
    print("Outliers:",out)
Zscore_outlier(redwine_df['total sulfur dioxide'])



#z method
#free sulfur dioxide
```

```
out=[]
def Zscore_outlier(df):
    m = np.mean(df)
    sd = np.std(df)
    for i in df:
        z = (i-m)/sd
        if np.abs(z) > 3:
            out.append(i)
    print("Outliers:",out)
Zscore_outlier(redwine_df['free sulfur dioxide'])
```

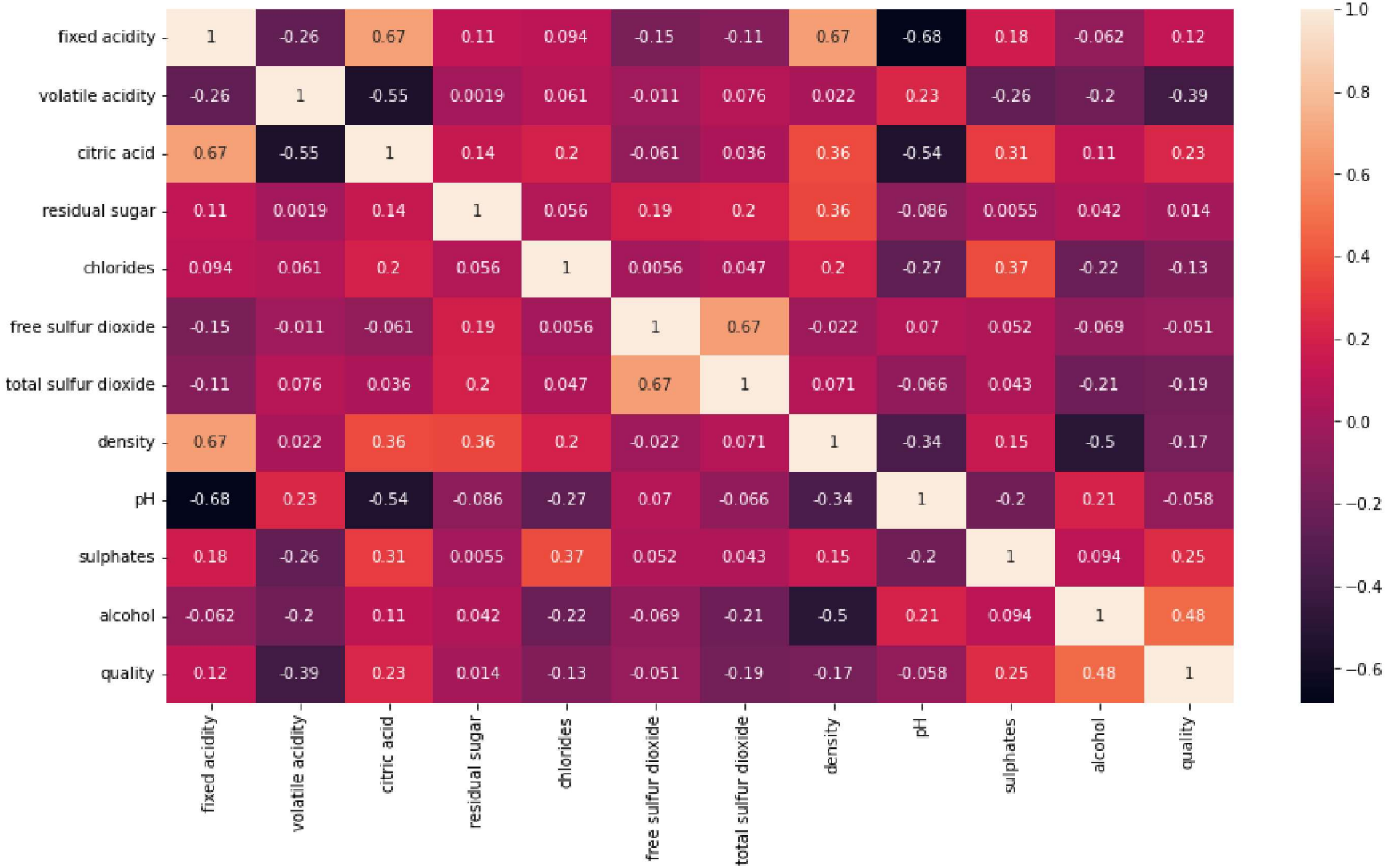
```
mean of the dataset is 46.46779237023139
std. deviation is 32.88503665178367
Outliers: [148.0, 153.0, 165.0, 151.0, 149.0, 147.0, 148.0, 155.0, 151.0, 152.0, 278.0, 289.0, 160.0, 147.0, 147.0]
Outliers: [52.0, 51.0, 50.0, 68.0, 68.0, 54.0, 53.0, 52.0, 51.0, 57.0, 50.0, 48.0, 48.0, 72.0, 51.0, 51.0, 52.0, 55.0, 5
5.0, 48.0, 48.0, 66.0]
```

In [14]:

```
plt.figure(figsize = (15, 8))

sns.heatmap(redwine_df.corr(), annot = True)
```

Out[14]:

```
<AxesSubplot:>
```



Explanatory Data Analysis (EDA)

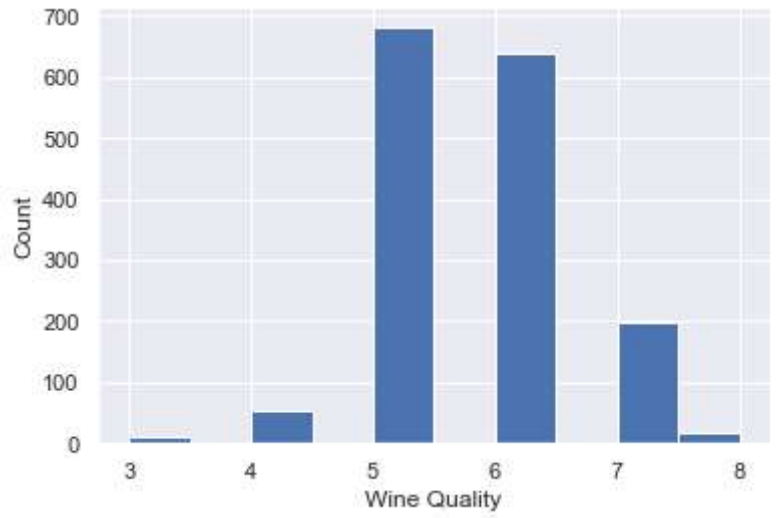
> Explorory Data Analysis recap

In [16]:

```
#Create histogram on 'quality' variable
sns.set()
redwine_df.quality.hist()
```

```
plt.xlabel('Wine Quality')
plt.ylabel('Count')
```

Out[16]:
```
Text(0, 0.5, 'Count')
```

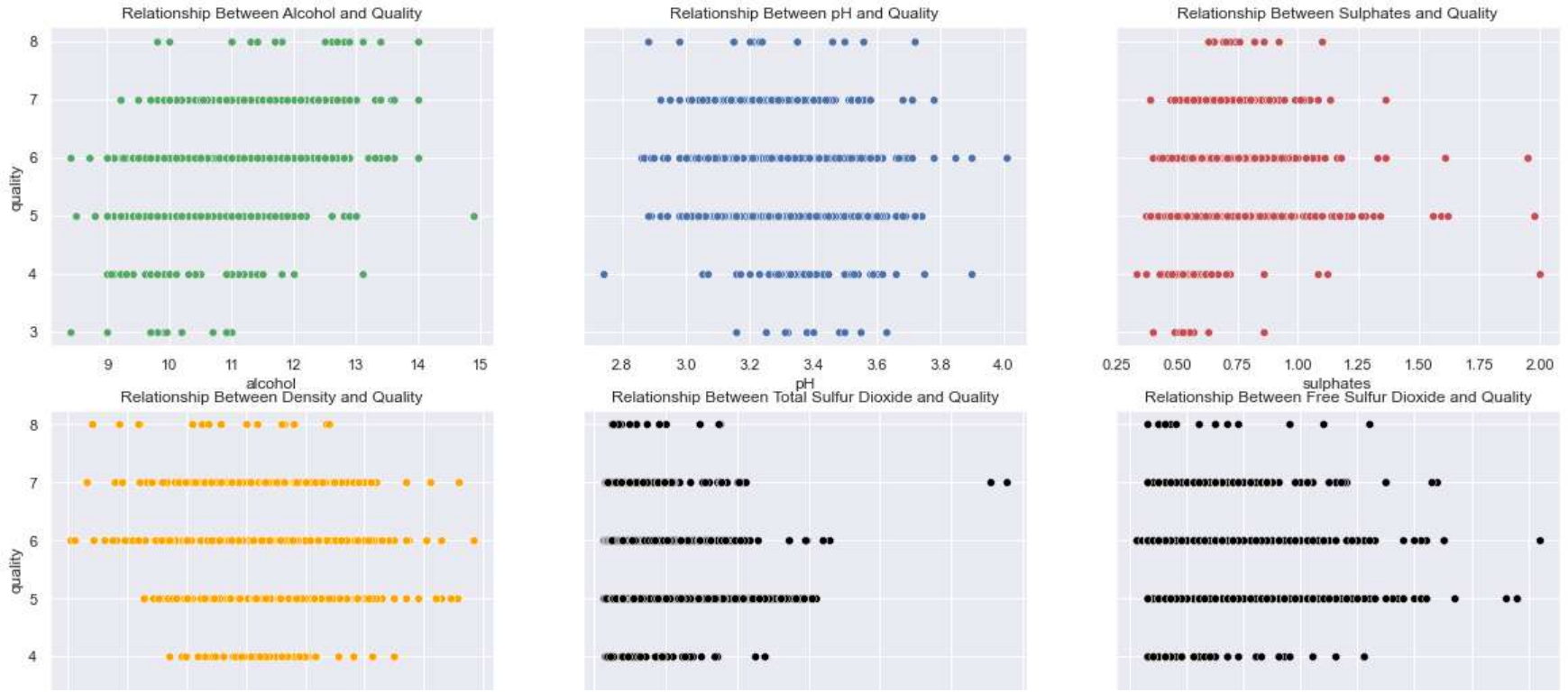


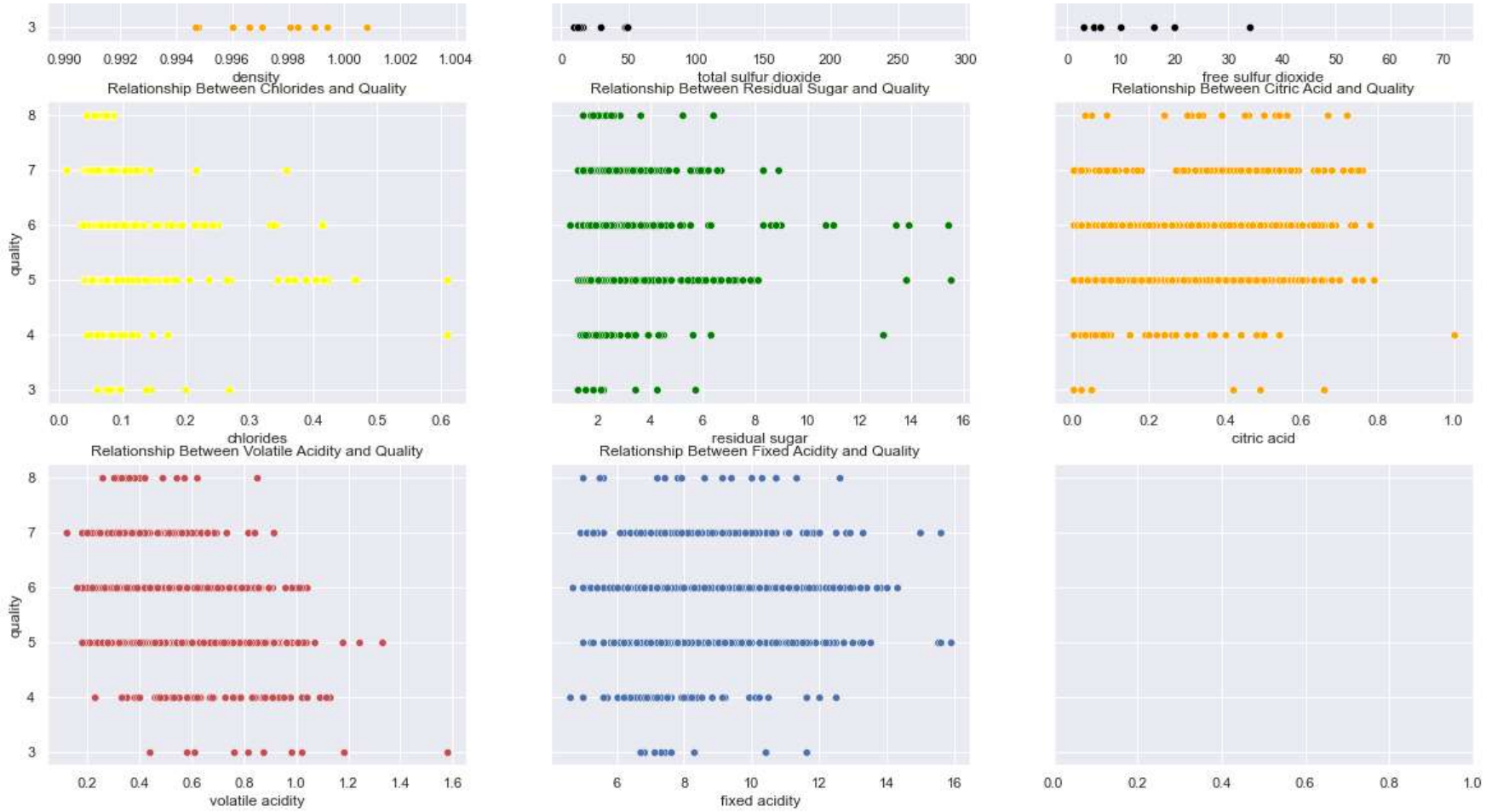
Explanation:

In [50]:
```
# Analyze the relationships between the predictors and the target variable ('quality').


fig, axes = plt.subplots(4, 3, figsize = (20,20), sharey = True)


sns.scatterplot(ax = axes[0,0], data = redwine_df, y = "quality", x = "alcohol", color = "g")
axes[0,0].set_title("Relationship Between Alcohol and Quality")

sns.scatterplot(ax = axes[0, 1], data = redwine_df, y = "quality", x = "pH", color = "b")
axes[0,1].set_title("Relationship Between pH and Quality")

sns.scatterplot(ax = axes[0, 2], data = redwine_df, y = "quality", x = "sulphates", color = "r")
axes[0,2].set_title("Relationship Between Sulphates and Quality")

sns.scatterplot(ax = axes[1,0], data = redwine_df, y = "quality", x = "density", color = "orange")
axes[1,0].set_title("Relationship Between Density and Quality")

sns.scatterplot(ax = axes[1,1], data = redwine_df, y = "quality", x = "total sulfur dioxide", color = "black")
```

```
axes[1,1].set_title("Relationship Between Total Sulfur Dioxide and Quality")

sns.scatterplot(ax = axes[1,2], data = redwine_df, y = "quality", x = "free sulfur dioxide", color = "black")
axes[1,2].set_title("Relationship Between Free Sulfur Dioxide and Quality")

sns.scatterplot(ax = axes[2,0], data = redwine_df, y = "quality", x = "chlorides", color = "yellow")
axes[2,0].set_title("Relationship Between Chlorides and Quality")

sns.scatterplot(ax = axes[2,1], data = redwine_df, y = "quality", x = "residual sugar", color = "green")
axes[2,1].set_title("Relationship Between Residual Sugar and Quality")

sns.scatterplot(ax = axes[2,2], data = redwine_df, y = "quality", x = "citric acid", color = "orange")
axes[2,2].set_title("Relationship Between Citric Acid and Quality")

sns.scatterplot(ax = axes[3,0], data = redwine_df, y = "quality", x = "volatile acidity", color = "r")
axes[3,0].set_title("Relationship Between Volatile Acidity and Quality")

sns.scatterplot(ax = axes[3,1], data = redwine_df, y = "quality", x = "fixed acidity", color = "b")
axes[3,1].set_title("Relationship Between Fixed Acidity and Quality")
```

Out[50]: Text(0.5, 1.0, 'Relationship Between Fixed Acidity and Quality')

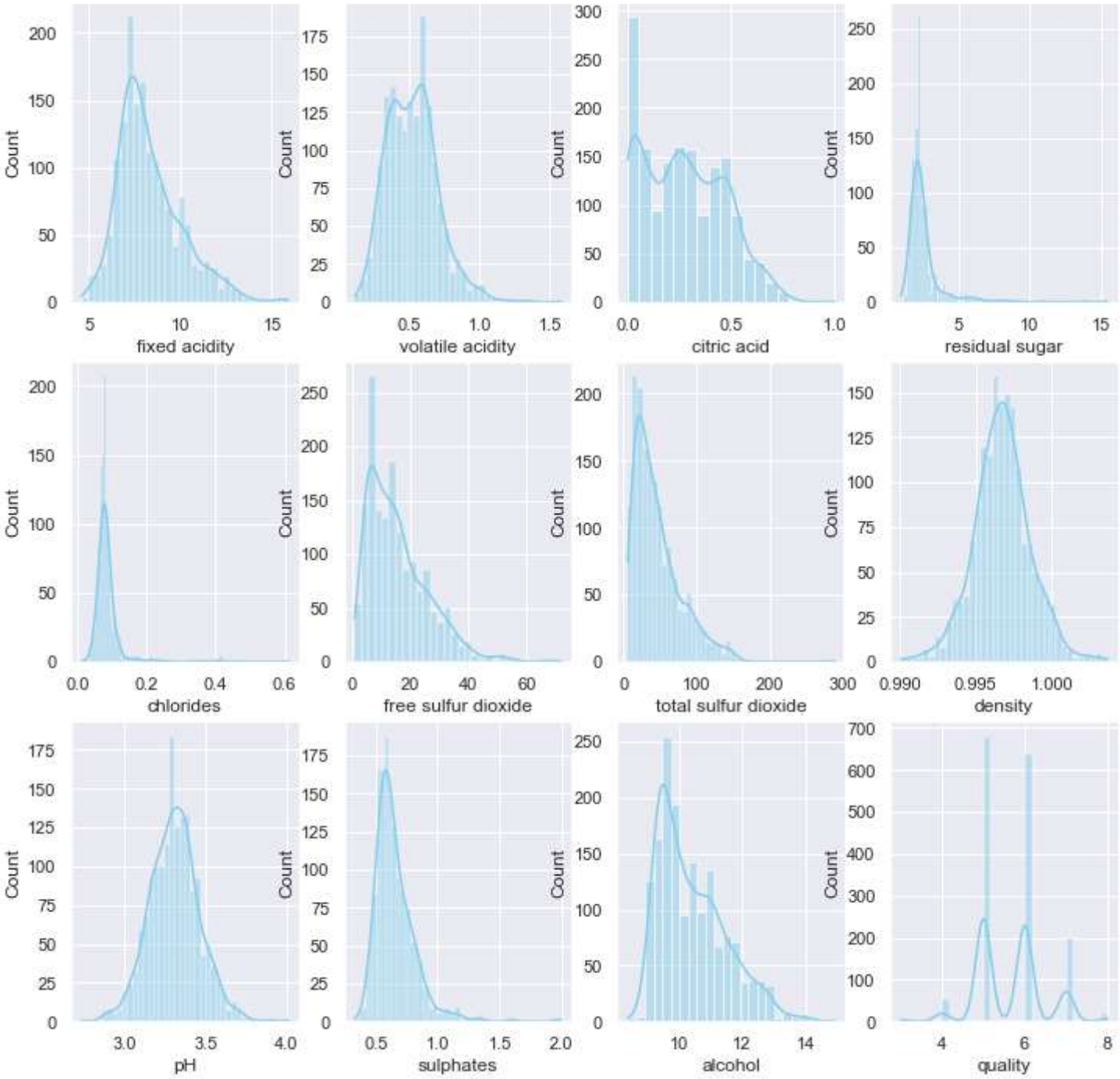




Explanation:

In [51]:

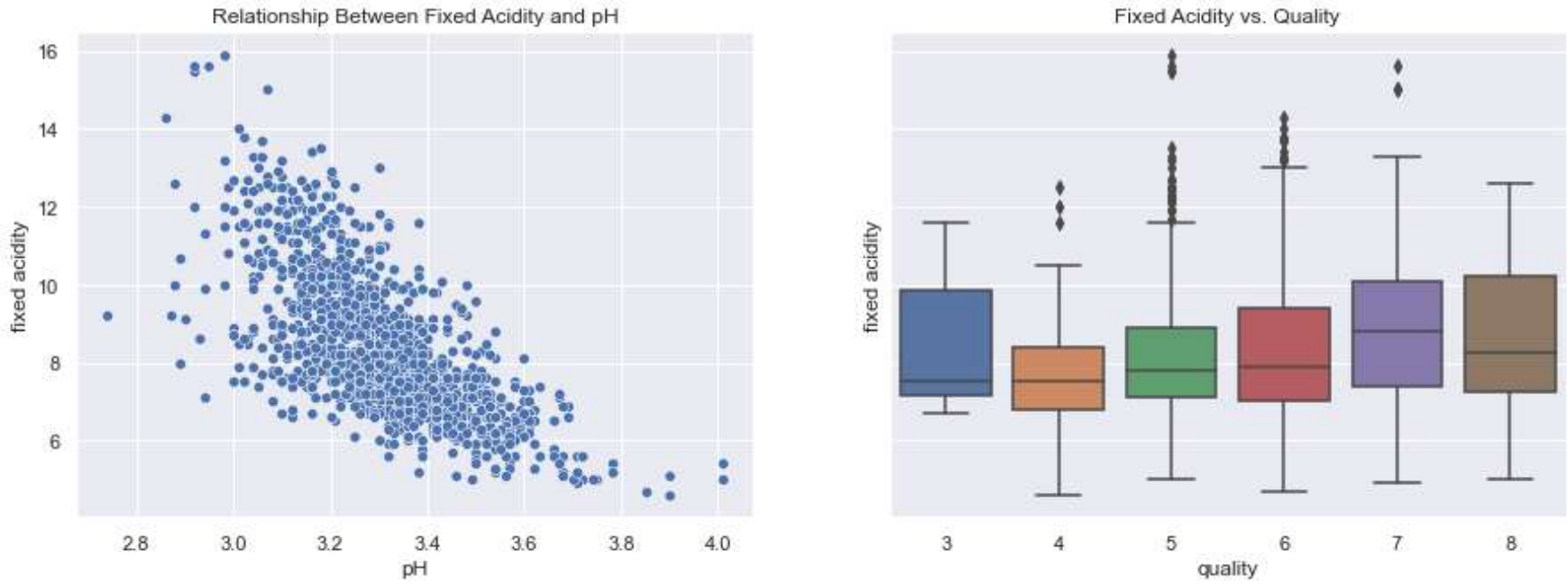
```
# Histogram

fig, axs = plt.subplots(3, 4, figsize=(12, 12))
columns = redwine_df.columns[:12]
k=0
sns.set(font_scale=1)
for i in range(3):
    for j in range(4):
        sns.histplot(data=redwine_df, x=columns[k], kde=True, color="skyblue", ax=axs[i, j])
        k+=1
```


Count
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality

Explanation:

In [54]:
```
fig, axes = plt.subplots(1, 2, figsize = (15, 5), sharey = True)

sns.scatterplot(ax = axes[0], data = redwine_df, y = "fixed acidity", x = "pH")
axes[0].set_title("Relationship Between Fixed Acidity and pH")

sns.boxplot(ax = axes[1], data = redwine_df, y = "fixed acidity", x = "quality")
axes[1].set_title("Fixed Acidity vs. Quality")
```

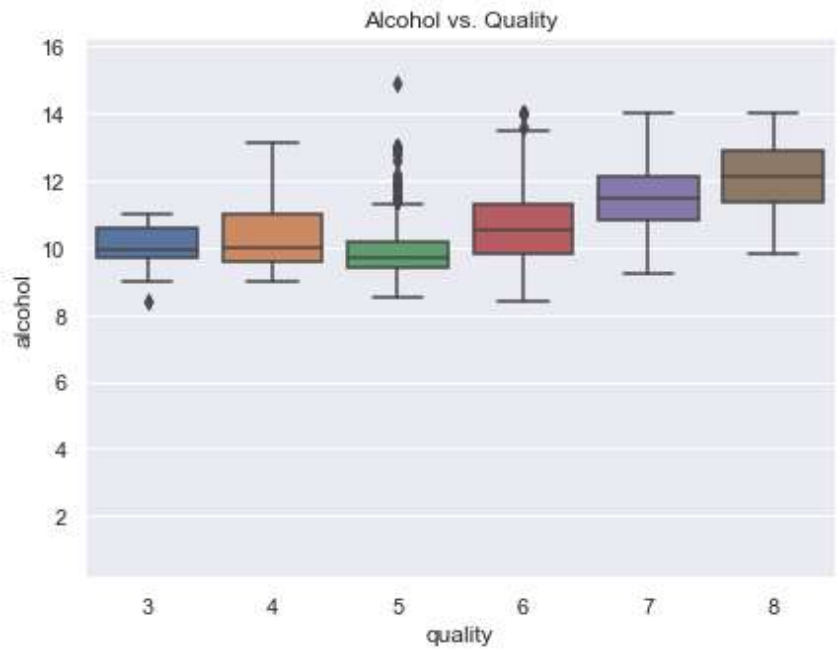
Out[54]:
```
Text(0.5, 1.0, 'Fixed Acidity vs. Quality')
```

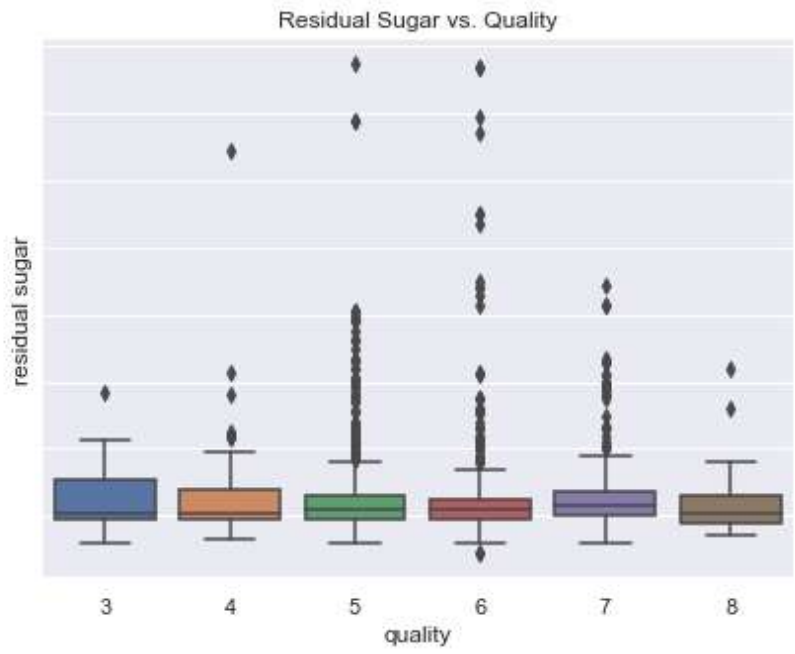


Explanation:

In [57]:
```
fig, axes = plt.subplots(1, 2, figsize = (15, 5), sharey = True)

sns.boxplot(ax = axes[0], data = redwine_df, y = "alcohol", x = "quality")
axes[0].set_title("Alcohol vs. Quality")

sns.boxplot(ax = axes[1], data = redwine_df, y = "residual sugar", x = "quality")
axes[1].set_title("Residual Sugar vs. Quality")
```

Out[57]: Text(0.5, 1.0, 'Residual Sugar vs. Quality')





Explanation

Data Splitting

In [159…

```
redwine_df.head()
```

Out[159…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [160…

```
for idx in redwine_df.index:
    if redwine_df["quality"][idx] <=5:
        redwine_df["quality"][idx] = 0

    if redwine_df["quality"][idx] >= 6:
        redwine_df["quality"][idx] = 1



redwine_df.head()
```

Out[160…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0

In [161…

```
y = redwine_df['quality'].to_numpy()
x = redwine_df.drop(columns=['quality'])
```

In [162…

```
#Standardize the dataset
scaler = preprocessing.StandardScaler()
x_norm = scaler.fit_transform(x * 1.0)
```

In [195…

```
#Split the full dataframe into 60/40.

train_x, test_x, train_y, test_y = train_test_split(x_norm, y, test_size=0.4, random_state=1)
train_x.shape, test_x.shape
```

Out[195…

```
((959, 11), (640, 11))
```

Data Modeling

LDA

In [ ]:

Gradient Boosting

In [ ]:

Logistic Regression

In [ ]:

Random Forest

In [ ]:

Decision Trees

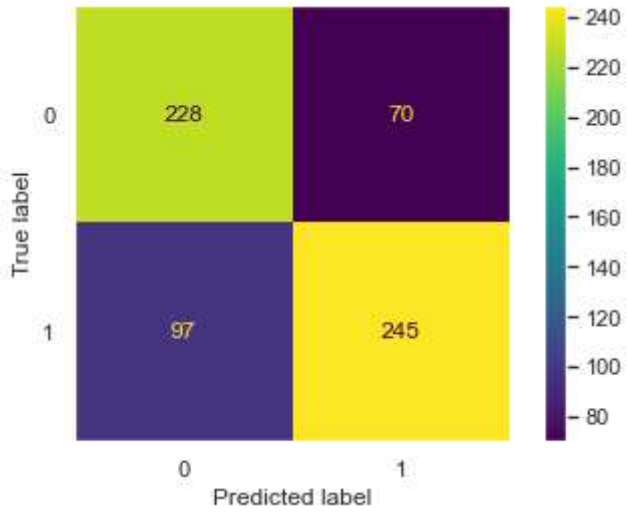
One of the aspects of decision trees that was relateable to our project goal was its ability to effectively choose between various actions. For this project, supervised learning would be the best way to go. The objective is to predict which wines are low quality (0) and high quality (1). Part of the process in determining the output is understanding how the output was chosen. By using decision tree, we can examine the choices made in the process and what role the predictors played. After fitting the data, predictions were made with the testing data. A confusion matrix was plotted along with a classification report showing how well the model did. The accuracy score for this model is 74 and the cross validation score is approx 73 percent. Recall gives a general overview saying of the total amount of actual good quality values, which were correctly predicted as good quality. Precision says of the total amount of predicted good quality values, how many actually have a true value of 6 or higher. F1 score focuses on recall and precision simultaneously. F1 score compare the performance of the two metrics.

In [200…

```
from sklearn import tree
from sklearn.metrics import classification_report
```

```
from sklearn.model_selection import cross_val_score
import statistics as stats

decisiontree = DecisionTreeClassifier().fit(train_x, train_y)

y_pred = decisiontree.predict(test_x)


plot_confusion_matrix(decisiontree, test_x, test_y)
plt.grid(False)

print(classification_report(test_y, y_pred))

print("Cross Val Score: ", stats.mean(cross_val_score(decisiontree, train_x, train_y, cv = 5)))
```

```
              precision    recall  f1-score   support

           0       0.70      0.77      0.73       298
           1       0.78      0.72      0.75       342

    accuracy                           0.74       640
   macro avg       0.74      0.74      0.74       640
weighted avg       0.74      0.74      0.74       640

Cross Val Score:  0.7267724694589878
```

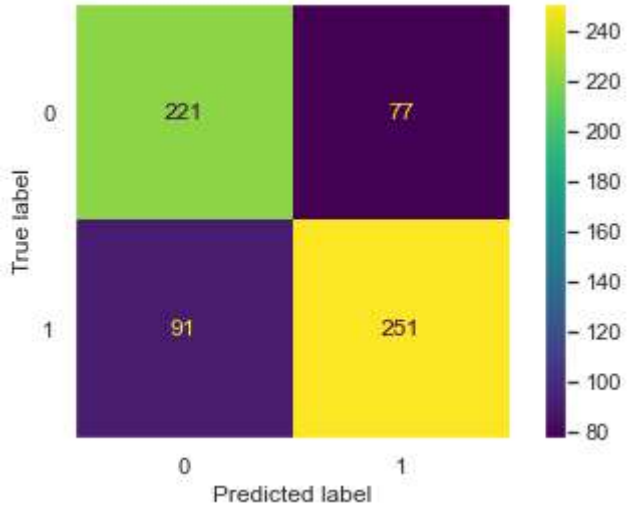


Neural Network

In [201…

```
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
import statistics as stats
from sklearn.neural_network import MLPClassifier
from dmba import classificationSummary


neuralnet = MLPClassifier(hidden_layer_sizes = (3), activation = 'logistic', solver = 'lbfgs', random_state =1).fit(trair

nnet_pred = neuralnet.predict(test_x)


plot_confusion_matrix(neuralnet, test_x, test_y)
plt.grid(False)

print(classification_report(test_y, nnet_pred))

print("Cross Val Score: ", stats.mean(cross_val_score(neuralnet, train_x, train_y, cv = 5)))
```

```
              precision    recall  f1-score   support

           0       0.71      0.74      0.72       298
           1       0.77      0.73      0.75       342

    accuracy                           0.74       640
   macro avg       0.74      0.74      0.74       640
weighted avg       0.74      0.74      0.74       640

Cross Val Score:  0.7184173211169285
```

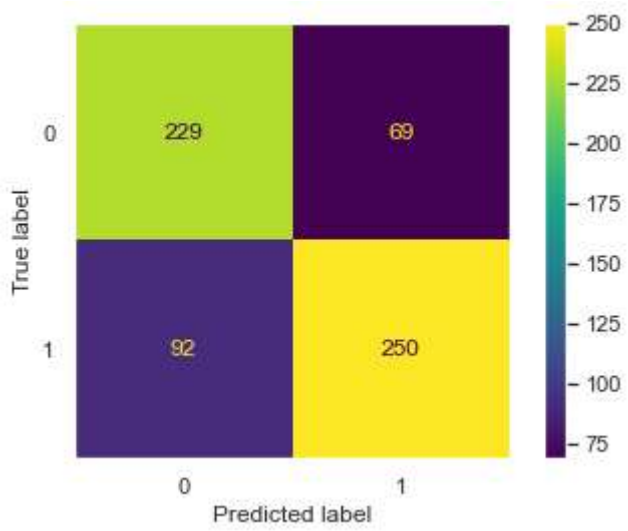


Support Vector Machines (SVM)

In [203…

```
from sklearn import svm

supportvec = svm.SVC().fit(train_x, train_y)

svm_pred = supportvec.predict(test_x)


plot_confusion_matrix(supportvec, test_x, test_y)
plt.grid(False)

print(classification_report(test_y, svm_pred))

print("Cross Val Score: ", stats.mean(cross_val_score(supportvec, train_x, train_y, cv = 5)))
```

```
              precision    recall  f1-score   support

           0       0.71      0.77      0.74       298
           1       0.78      0.73      0.76       342

    accuracy                           0.75       640
   macro avg       0.75      0.75      0.75       640
weighted avg       0.75      0.75      0.75       640

Cross Val Score:  0.7622273123909249
```

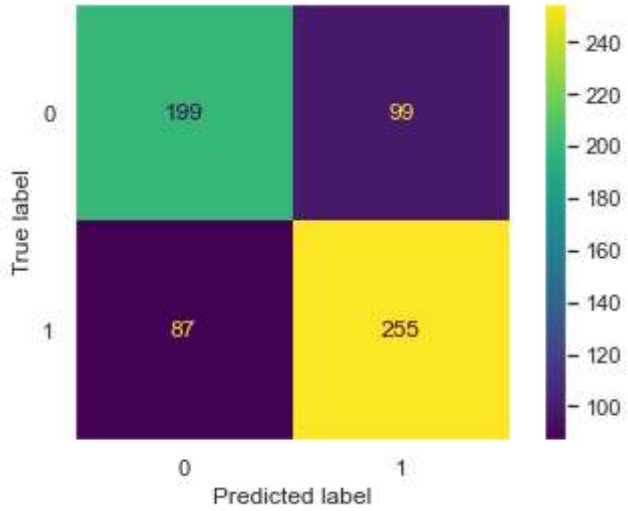


K-Nearest Neighbors (KNN)

In [204…

```
kneighbors = KNeighborsClassifier().fit(train_x, train_y)

knn_pred = kneighbors.predict(test_x)


plot_confusion_matrix(kneighbors, test_x, test_y)
plt.grid(False)

print(classification_report(test_y, knn_pred))

print("Cross Val Score: ", stats.mean(cross_val_score(kneighbors, train_x, train_y, cv = 5)))
```

```
              precision    recall  f1-score   support

           0       0.70      0.67      0.68       298
           1       0.72      0.75      0.73       342

    accuracy                           0.71       640
   macro avg       0.71      0.71      0.71       640
weighted avg       0.71      0.71      0.71       640

Cross Val Score:  0.7038612565445026
```


199
99
87
255
0
1
True label
0
1
Predicted label
240
220
200
180
160
140
120
100