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 [63]:

```
#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
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split, cross_val_score
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
!pip install statsmodels
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, classification_report
import seaborn as sns
from sklearn.preprocessing import StandardScaler

warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model
```

```
from sklearn.preprocessing import LabelEncoder
import matplotlib.pylab as plt
import dmba
from dmba import regressionSummary
from dmba import adjusted_r2_score, AIC_score, BIC_score
```

```
Requirement already satisfied: statsmodels in c:\users\andre\.conda\anaconda3\lib\site-pac
kages (0.13.2)
Requirement already satisfied: patsy>=0.5.2 in c:\users\andre\.conda\anaconda3\lib\site-pa
ckages (from statsmodels) (0.5.3)
Requirement already satisfied: numpy>=1.17 in c:\users\andre\.conda\anaconda3\lib\site-pac
kages (from statsmodels) (1.21.5)
Requirement already satisfied: pandas>=0.25 in c:\users\andre\.conda\anaconda3\lib\site-pa
ckages (from statsmodels) (1.4.1)
Requirement already satisfied: scipy>=1.3 in c:\users\andre\.conda\anaconda3\lib\site-pack
ages (from statsmodels) (1.7.3)
Requirement already satisfied: packaging>=21.3 in c:\users\andre\.conda\anaconda3\lib\site
-packages (from statsmodels) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\andre\.conda\anaconda3
\lib\site-packages (from packaging>=21.3->statsmodels) (3.0.4)
Requirement already satisfied: pytz>=2020.1 in c:\users\andre\.conda\anaconda3\lib\site-pa
ckages (from pandas>=0.25->statsmodels) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\andre\.conda\anaconda3\l
ib\site-packages (from pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: six in c:\users\andre\.conda\anaconda3\lib\site-packages (f
rom patsy>=0.5.2->statsmodels) (1.16.0)
```

In [26]:

```
#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[26]:

	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

Data Preprocessing

> Data Preprocessing Explanation for this section -> The uploaded wine dataset was preprocessed, which involved evaluating any necessary modifications needed, from outliers, correlations, or missing values, towards the dataset as a prepatory procedure for the final model. The shape of the dataset features 1,599 entries with 12 columns with no missing data detected. Since the objective is to develop a model that predicts the quality of the wine, the 'quality' predictor was designated as the target variable. There were six unique elements of an array (values 3 to 8) within the 'quality' predictor, and each element served as a scale to rate the quality of the wine. The next

procedure made to the dataset was to detect any outliers within each predictor using the Z method, which also calculated the predictors' mean and standard deviation. The dataset contained many outliers, with 'total sulfur dioxide' predictor containing the most. Lastly, a heatmap was created to assess the correlations of the predictors. The 'density' predictor shared a strong positive correlation value of 0.67 with 'fixed acidity' and 'citric acid'. On the contrary, 'fixed acidity' and 'pH' shared a strong negative correlation value of -0.68.

In [27]:

```
# 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 [28]:

```
redwine_df.describe()
```

Out[28]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	

In [29]:

```
# Check for data size
redwine_df.shape
```

Out[29]:

```
(1599, 12)
```

In [30]:

```
redwine_df['quality'].unique()
```

Out[30]:

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

In [31]:

```
redwine_df.isna().sum()
```

Out[31]:

```
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 [32]:

```
redwine_df.dtypes
```

Out[32]:

```
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 [33]:

```
# 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'])
```

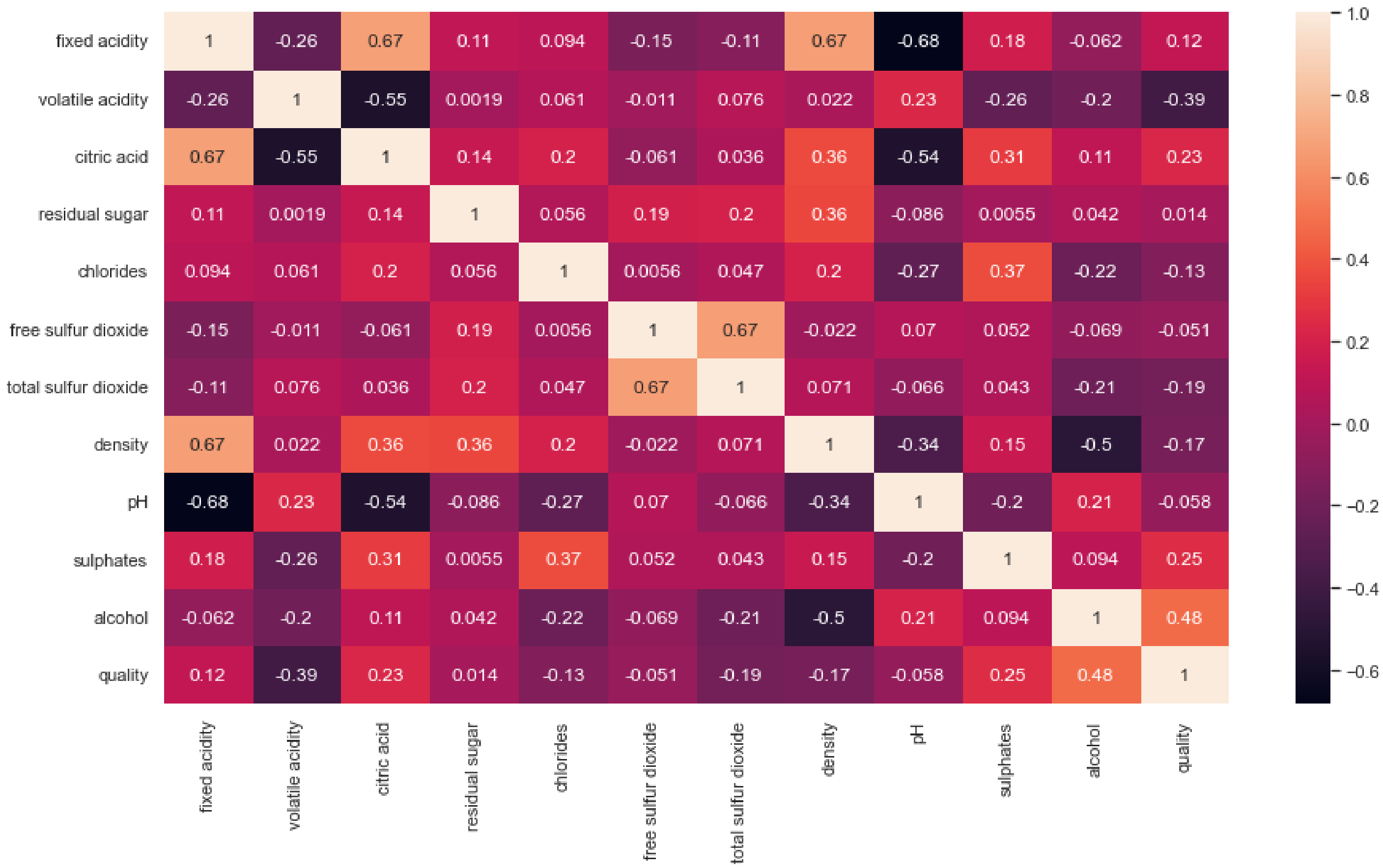
```
mean of the dataset is 46.46779237023139
std. deviation is 32.88503665178374
Outliers: [148.0, 153.0, 165.0, 151.0, 149.0, 147.0, 148.0, 155.0, 151.0, 152.0, 278.0, 28
9.0, 160.0, 147.0, 147.0]
Outliers: [52.0, 51.0, 50.0, 68.0, 68.0, 54.0, 53.0, 52.0, 51.0, 57.0, 50.0, 48.0, 48.0, 7
2.0, 51.0, 51.0, 52.0, 55.0, 55.0, 48.0, 48.0, 66.0]
```

In [34]:

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

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

Out[34]:

```
<AxesSubplot:>
```



Explanatory Data Analysis (EDA)

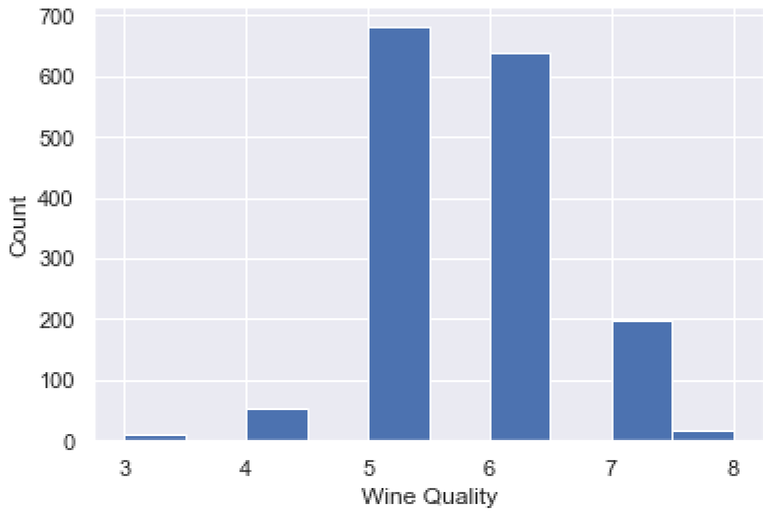
> Explorory Data Analysis recap: Exploratory Data Analysis (EDA) was implemented to evaluate the qualities of each predictor within the dataset. This procedure involved generating a number of visualizations to identify certain trends of each predictor from the dataset, test hypotheses, and evaluate assumptions

In [35]:

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

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

Out[35]: Text(0, 0.5, 'Count')



In [36]:

```python
redwine_df['quality'].value_counts()
```

Out[36]:

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

> Explanation: The first visualization made above was a histogram, which highlights the distributed quantity of the targeted predictor 'Wine Quality' since the objective is to create a model that predicts the quality of the wine as good or bad. It was found that the average quality-type wines (rated 5 or 6) generated the highest count, which also indicates that they were most distributed among businesses. The value count function above specifies the value of wines that fit each category, which they confirm the average-rated wines being distributed the most.

In [ ]:

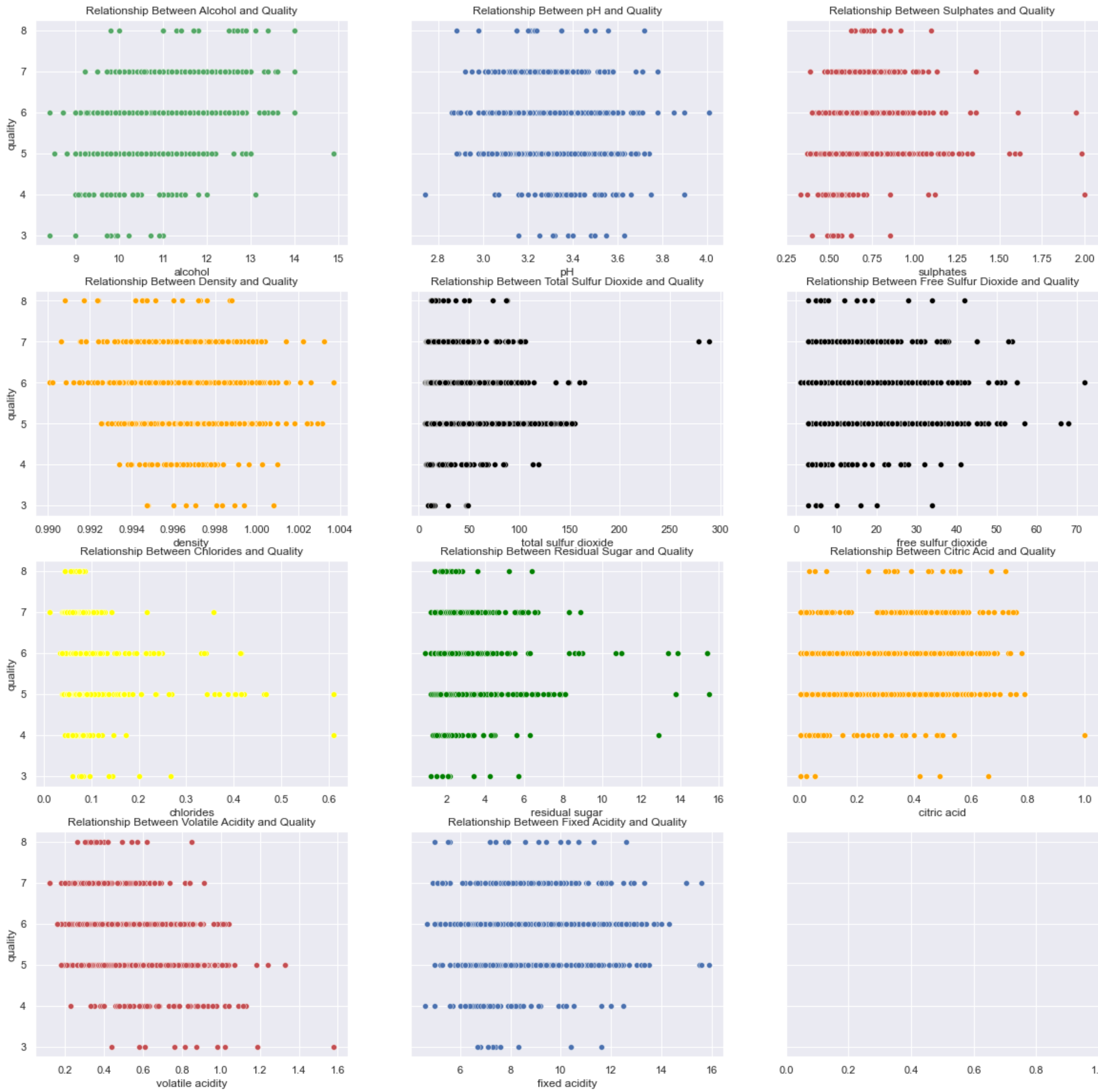
In [37]:

```python
# 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 = '
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
axes[0,2].set_title("Relationship Between Sulphates and Quality")

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

```
sns.scatterplot(ax = axes[1,1], data = redwine_df, y = "quality", x = "total sulfur dioxic
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
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 =
axes[2,0].set_title("Relationship Between Chlorides and Quality")

sns.scatterplot(ax = axes[2,1], data = redwine_df, y = "quality", x = "residual sugar", co
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
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",
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", col
axes[3,1].set_title("Relationship Between Fixed Acidity and Quality")
```

Out[37]:

```
Text(0.5, 1.0, 'Relationship Between Fixed Acidity and Quality')
```

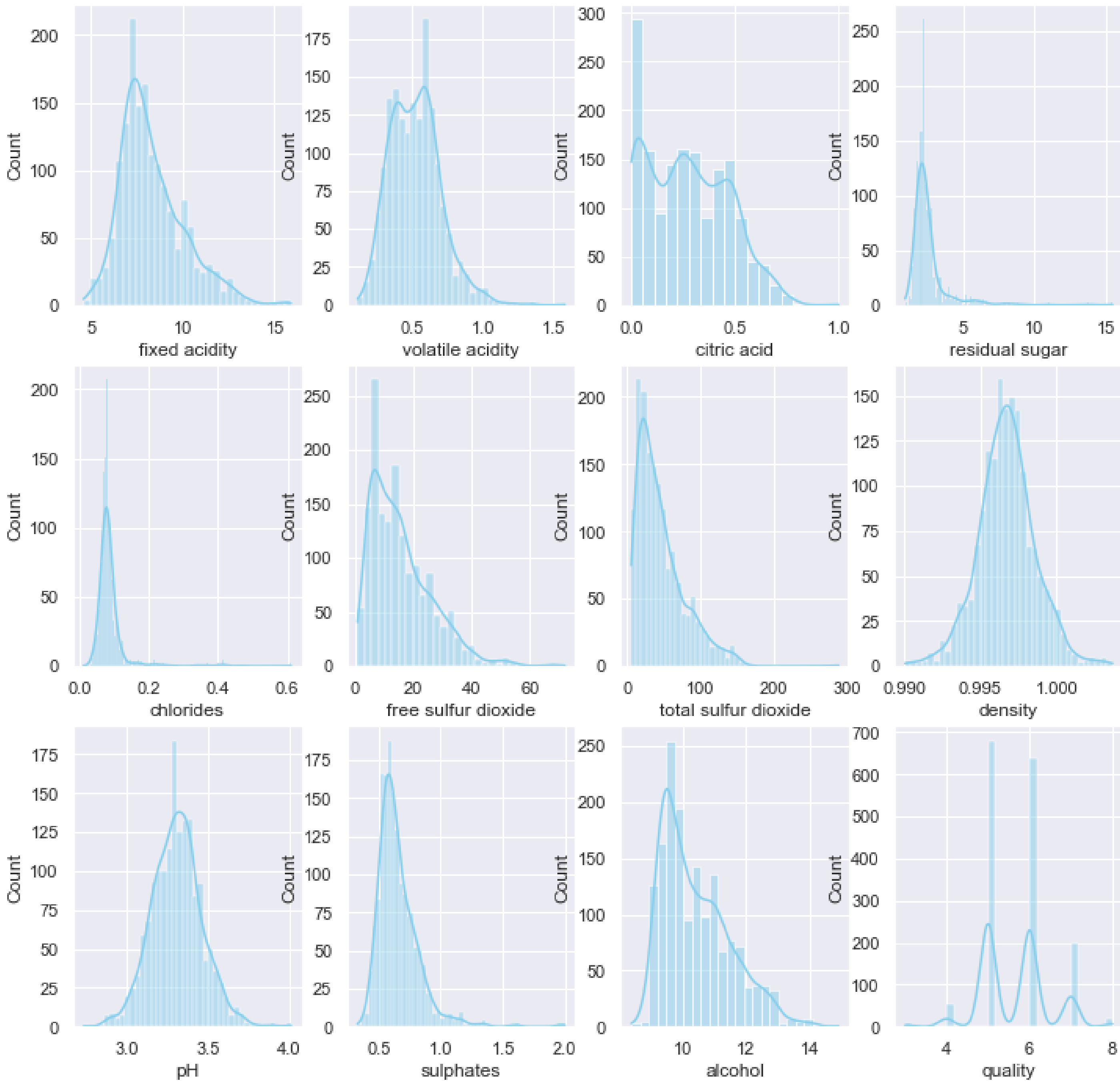


> Explanation: The next step was to analyze the relationships between the predictors and 'quality'. Using scatterplots to distribute a visual representation of each predictor versus 'quality', all predictors were found to have a strong relationship between the average-rated wine qualities. From the visualizations, the quality-type wines that were rated 5 or 6 garnered most of each predictor. The scatterplots also indicated a strong presence of outliers within certain predictors that includes sulfur dioxide, residual sugar, and chlorides.

In [38]:

```
# 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,
    k+=1
```



> Explanation: As previously done to the target variable 'quality', the remaining predictors from the dataset were visualized using histograms. A distribution plot was also distributed to evaluate the distribution patterns of each predictor in a convenient form and depict any outliers within each predictor. From the visualizations, the 'sulfur dioxide' and 'acid' predictors contained the largest amount of outliers. In addition, nearly all the predictors are skewed positively or negatively.

In [ ]:

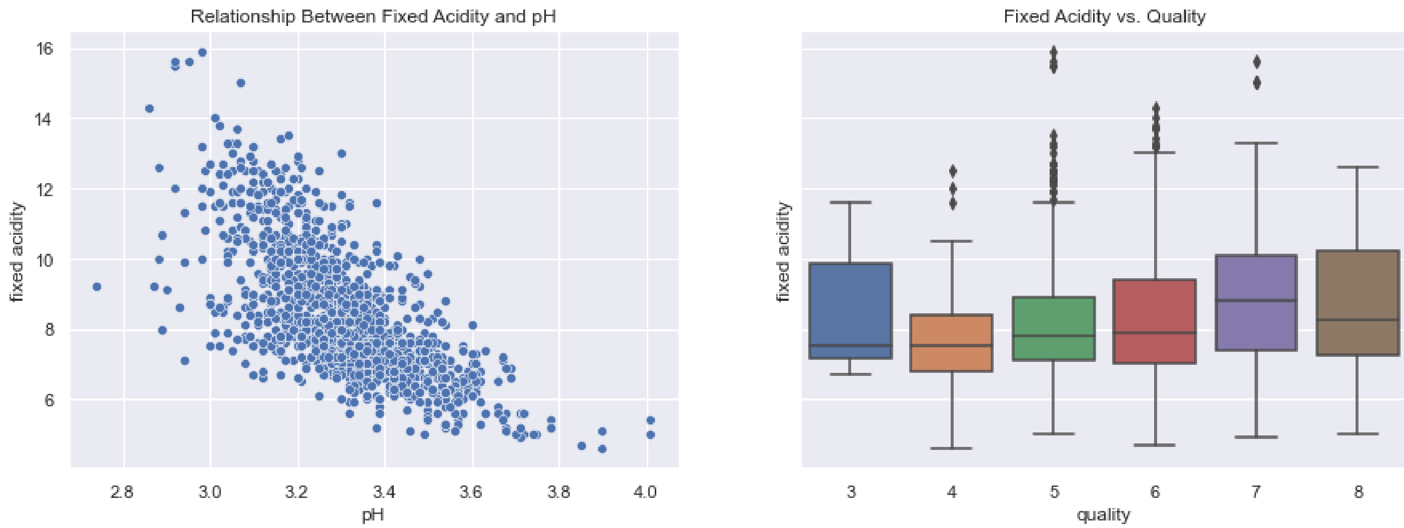
```

```

In [39]:

```
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")
```

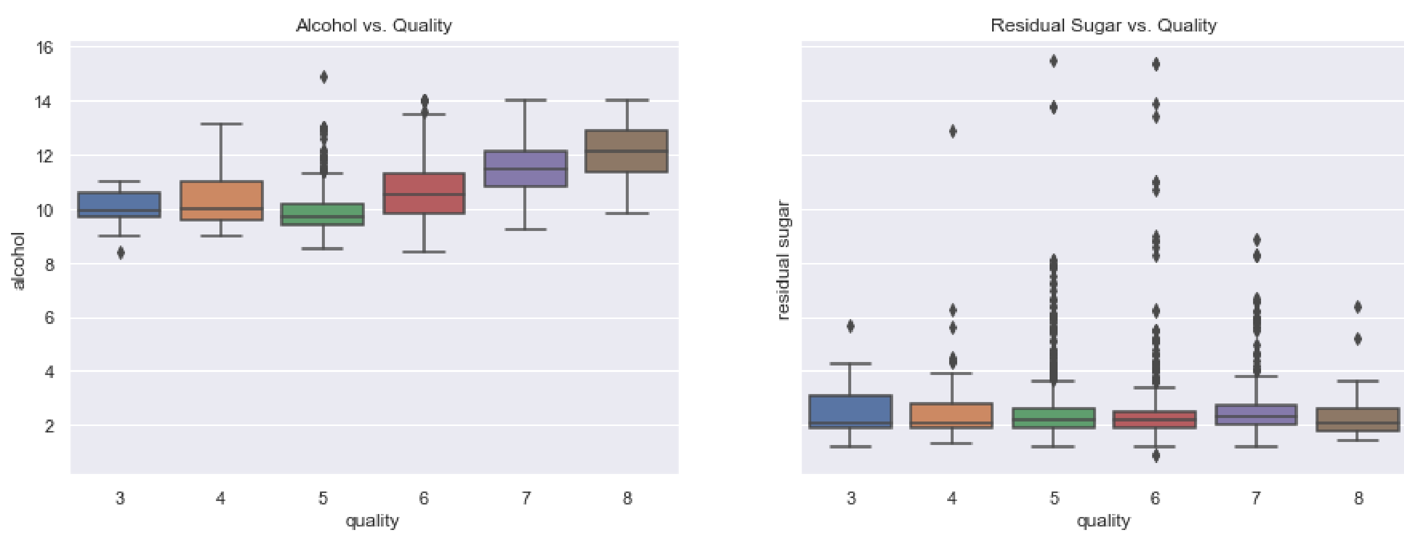
Out[39]: Text(0.5, 1.0, 'Fixed Acidity vs. Quality')



> Explanation: The following visuals above assessed the relationship between 'fixed acidity' to 'pH' and 'quality'. For the scatterplot on the left, 'fixed acidity' and 'pH' share a strong negative correlation, where a decrease in 'fixed acidity' leads to an increse in 'pH'. For the boxplot, a majority of the wine qualities lean towards the distribution of 'fixed' acidity' and 'quality' being positively skewed. Outliers are clearly visualized for each wine quality type when assessing the relationship between 'fixed acidity' and 'quality'.

In [40]:
```
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[40]: Text(0.5, 1.0, 'Residual Sugar vs. Quality')



> Explanation: The visuals above feature boxplots that depicts the relationship of 'pH' to 'alcohol' and 'residual sugar'. The high-rated wine qualities that contain alcohol feature a distribution that is normally distributed while the low-rated wine qualities with alcohol are more skewed to the left. Heavy presence of outliers were detected while evaluating the relationship between 'residual sugar' and 'quality'.

In [ ]:

In [44]:

```
redwine_df['good_quality'] = ["yes" if x>=7 else 'no' for x in redwine_df['quality']]
redwine_df.head(20)
```

Out[44]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	good_qu
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	
5	7.4	0.660	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5	
6	7.9	0.600	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5	
7	7.3	0.650	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7	
8	7.8	0.580	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7	
9	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5	
10	6.7	0.580	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	9.2	5	
11	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5	
12	5.6	0.615	0.00	1.6	0.089	16.0	59.0	0.9943	3.58	0.52	9.9	5	
13	7.8	0.610	0.29	1.6	0.114	9.0	29.0	0.9974	3.26	1.56	9.1	5	
14	8.9	0.620	0.18	3.8	0.176	52.0	145.0	0.9986	3.16	0.88	9.2	5	
15	8.9	0.620	0.19	3.9	0.170	51.0	148.0	0.9986	3.17	0.93	9.2	5	
16	8.5	0.280	0.56	1.8	0.092	35.0	103.0	0.9969	3.30	0.75	10.5	7	
17	8.1	0.560	0.28	1.7	0.368	16.0	56.0	0.9968	3.11	1.28	9.3	5	
18	7.4	0.590	0.08	4.4	0.086	6.0	29.0	0.9974	3.38	0.50	9.0	4	
19	7.9	0.320	0.51	1.8	0.341	17.0	56.0	0.9969	3.04	1.08	9.2	6	

In [47]:

```
LE = LabelEncoder()
redwine_df['good_quality']=LE.fit_transform(redwine_df['good_quality'])
```

In [48]:

```
redwine_df['good_quality'].value_counts()
```

Out[48]:

```
0    1382
1     217
```

```
Name: good_quality, dtype: int64
```

For this next procedure, the wines with a quality rating of 7 or above were labeled to be in good quality while anything below 7 were labeled to be in bad quality. Using LabelEncoder to accumulate and categorize the good quality wines from the bad ones, the dataset contained more bad quality-type wines than good wines.

In [ ]:

```

```

Data Splitting

In [49]:

```
#Set target variable to y and the remaining predictors to x
y = redwine_df['quality'].to_numpy()
x = redwine_df.drop(columns=['quality'])
```

In [50]:

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

In [55]:

```
#Split the full dataframe into 60/40.
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.4, random_state=1)
train_x.shape, valid_x.shape
```

Out[55]:

```
((959, 12), (640, 12))
```

In [56]:

```
#Scaling independent variables
sc = StandardScaler()
sc.fit_transform(train_x)
sc.transform(test_x)
```

Out[56]:

```
array([[ 0.25793474, -0.67886115,  1.85511303, ..., -0.02249274,
         0.06449435, -0.40126917],
       [ 0.20199638,  0.5535626 ,  0.04467877, ..., -0.19233951,
        -0.21771133, -0.40126917],
       [ 1.15294845, -1.07099598,  1.55337399, ...,  0.20396962,
         0.81704282, -0.40126917],
       ...,
       [-0.24551047, -1.23905376,  0.24583814, ..., -0.58864864,
         2.03993409, -0.40126917],
       [-1.36427761, -1.35109229, -0.10619075, ..., -0.70187982,
        -0.12364277, -0.40126917],
       [ 0.87325666,  0.32948555, -1.01140788, ...,  0.26058521,
        -0.59398557, -0.40126917]])
```

Data Modeling

LDA

In [ ]:

```

```

Gradient Boosting

In [ ]:

```

```

Logistic Regression

The logistic regression model was chosen to statistically predict the probability of a wine quality being good or bad. Using the model to generate the accuracy score, the test results generated an accuracy rating of 71.72%. The classification report of this model highlights that wines with a quality rating of 7 generated the highest values in precision (0.93), recall (1.00), and the f1 score (0.96). These scores indicate that the logistic regression model was effective in deciphering which wines were good or bad in quality.

In [57]:
```
Logit_reg = LogisticRegression()
Logit_reg.fit(train_x, train_y)
pred_y = Logit_reg.predict(test_x)
```

In [58]:
```
Logitreg_acc = accuracy_score(test_y, pred_y)
print("Accuracy Score:", Logitreg_acc)
```

```
Accuracy Score: 0.7171875
```

In [59]:
```
print(classification_report(test_y, pred_y))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        25
           5       0.68      0.76      0.72       271
           6       0.69      0.68      0.69       258
           7       0.93      1.00      0.96        78
           8       0.00      0.00      0.00         6

    accuracy                           0.72       640
   macro avg       0.38      0.41      0.39       640
weighted avg       0.68      0.72      0.70       640
```

In [64]:
```
#Cross Validation Score of logistic regression
LR_score = cross_val_score(Logit_reg, train_x, train_y, cv=10)
CV_LR = LR_score.mean().round(5)*100
print("Mean Score:", CV_LR,'%')
```

```
Mean Score: 70.911 %
```

In [ ]:

Random Forest

The random forest algorithm was chosen to classify and predict the outcome of a wine quality being good or bad. This model generated an accuracy score rating of 75.78%. The classification report of this model highlights that wines with a quality rating of 7 generated the highest values in precision (0.93), recall (0.99), and the f1 score (0.96). These scores indicate that the logistic regression model was effective in deciphering which wines were good or bad in quality.

In [60]:
```
rf = RandomForestClassifier()
rf.fit(train_x, train_y)
rf_pred = rf.predict(test_x)
```

In [61]:

```
rf_acc = accuracy_score(test_y, rf_pred)
print("Accuracy Score:", rf_acc)
```

```
Accuracy Score: 0.7578125
```

In [62]:

```
print(classification_report(test_y, rf_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        25
           5       0.72      0.79      0.76       271
           6       0.75      0.75      0.75       258
           7       0.93      0.99      0.96        78
           8       0.00      0.00      0.00         6

    accuracy                           0.76       640
   macro avg       0.40      0.42      0.41       640
weighted avg       0.72      0.76      0.74       640
```

In [65]:

```
#Cross Validation Score of random forest
RF_score = cross_val_score(rf, train_x, train_y, cv=10)
CV_RF = RF_score.mean().round(5)*100
print("Mean Score:", CV_RF,'%')
```

```
Mean Score: 75.809 %
```

In [ ]:

Decision Trees

In [ ]:

K-Nearest Neighbors (KNN) [Might change]

In [ ]:

In [ ]:

In [ ]: