CSC 767 Neural Networks & Deep Learning
Homework # 1
Part 3

Team Members:

Loading data and summarize data

```
In [20]: # Load libraries
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
import numpy as np
import seaborn as sns
```

```
In [21]: # Load dataset
url = 'winequality-white.csv'
names = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides',
         'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
         'quality']
dataset = read_csv(url, sep=';')
```

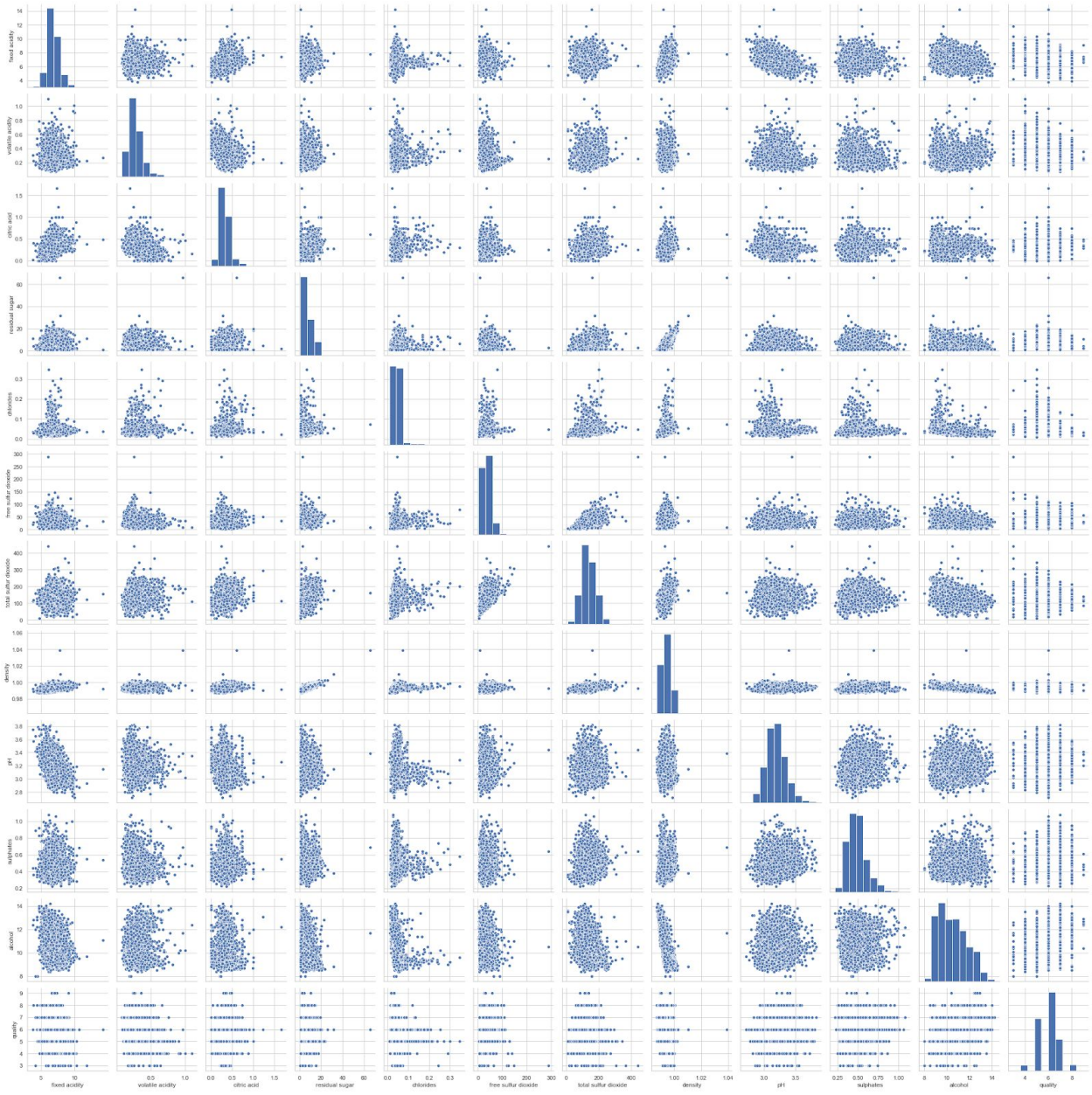
Listing 8: Complete any 8 calculations and plottings using seaborn package

Note: we did a few more than 8 because some plots are similar or repeat plots from part 1 or 2, just done with Seaborn. So we included some unique plots that we only did with Seaborn in part 3.

1. Pairplot Scatter

```
In [22]: # Part 3
# Listing 8
# inline plotting instead of popping out
%matplotlib inline

sns.set(style='whitegrid', context='notebook')
# Check correleation between the variables using Seaborn's pairplot.
sns.pairplot(dataset, height=2.5)
pyplot.tight_layout()
pyplot.savefig('seaborn-scatter1.png', dpi=300)
pyplot.show()
```



Analysis

Similar plots as in previous parts. This just helps visually the data and how attributes relate with each other and if any noticeable trends are visible. For the most part, concentrations of data are scattered and trends are not visible without more analysis.

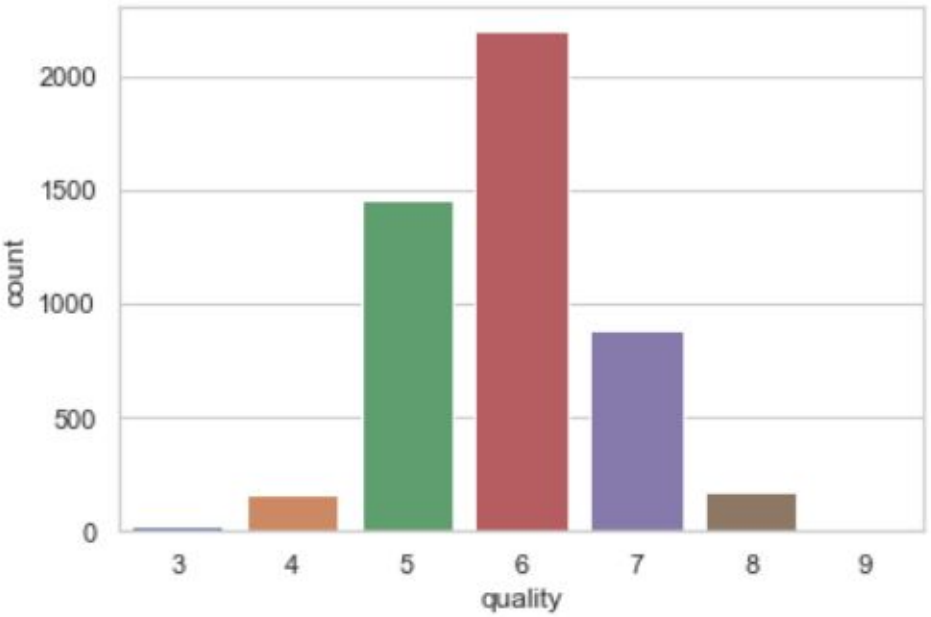
2, Count and graph of the target classification variable

```
In [7]: from collections import Counter
        Counter(dataset['quality'])

Out[7]: Counter({6: 2198, 5: 1457, 7: 880, 8: 175, 4: 163, 3: 20, 9: 5})

In [10]: # Count of the target variable
         sns.countplot(x='quality', data=dataset)

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x123bb2bdef0>
```



Analysis

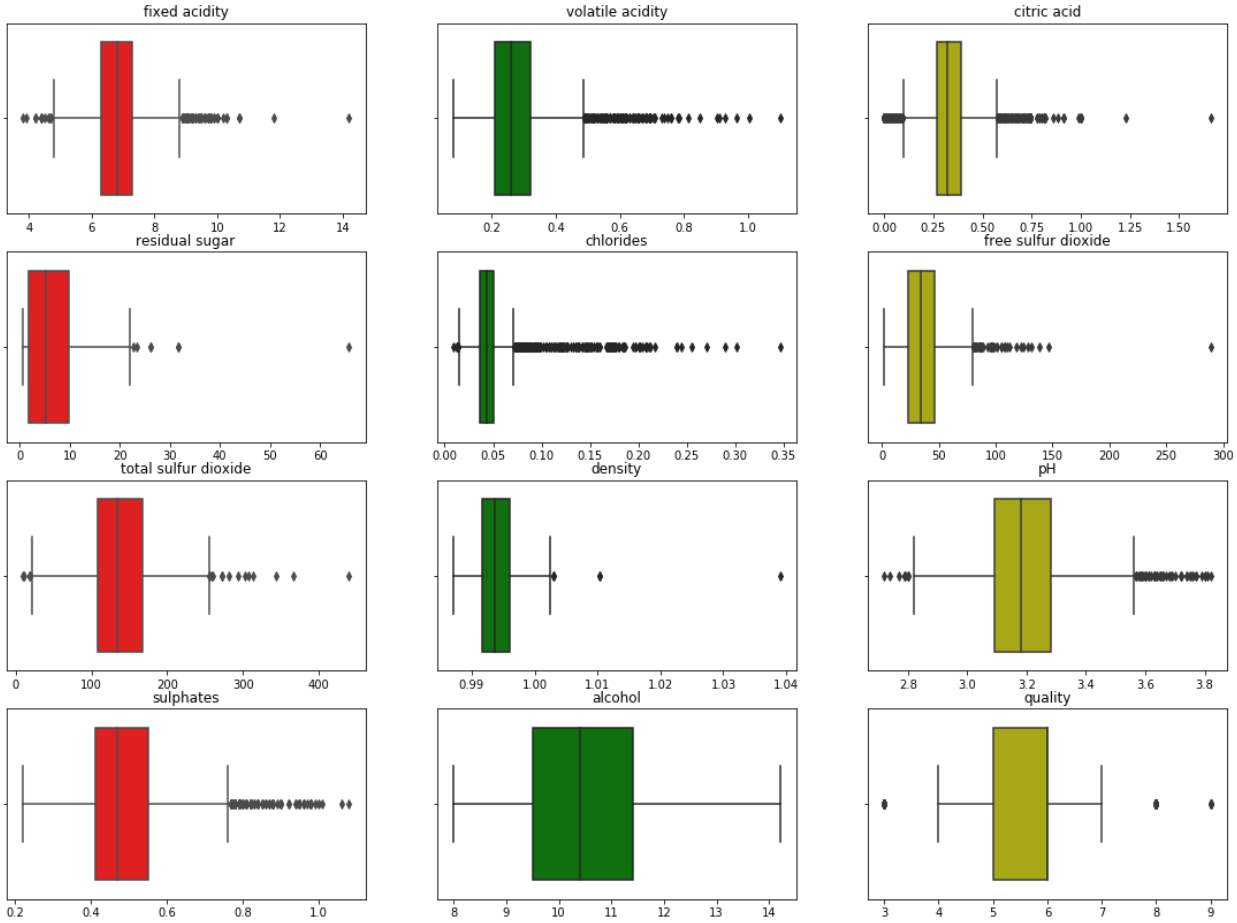
While quality varies between 3-7, we see that the vast majority of data points have qualities of 5-7. Perhaps limiting our data to the 3 qualities of 5,6,7 would have been ideal for our calculations and plots, for easier legibility. For example, our scatter plots and relationship plots show a larger range of qualities (3-7) which makes them more difficult to read and analyze.

3. Box and Whisker plot on each attributes

In [101]:
```
#box plot

f, axes = plt.subplots(4, 3, figsize=(19, 14))

sns.boxplot(dataset_only[:,0],color="r",ax=axes[0, 0]).set(title = 'fixed acidity')
sns.boxplot(dataset_only[:,1],color="g",ax=axes[0, 1]).set(title = 'volatile acidity')
sns.boxplot(dataset_only[:,2],color="y",ax=axes[0, 2]).set(title = 'citric acid')
sns.boxplot(dataset_only[:,3],color="r",ax=axes[1, 0]).set(title = 'residual sugar')
sns.boxplot(dataset_only[:,4],color="g",ax=axes[1, 1]).set(title = 'chlorides')
sns.boxplot(dataset_only[:,5],color="y",ax=axes[1, 2]).set(title = 'free sulfur dioxide')
sns.boxplot(dataset_only[:,6],color="r",ax=axes[2, 0]).set(title = 'total sulfur dioxide')
sns.boxplot(dataset_only[:,7],color="g",ax=axes[2, 1]).set(title = 'density')
sns.boxplot(dataset_only[:,8],color="y",ax=axes[2, 2]).set(title = 'pH')
sns.boxplot(dataset_only[:,9],color="r",ax=axes[3, 0]).set(title = 'sulphates')
sns.boxplot(dataset_only[:,10],color="g",ax=axes[3, 1]).set(title = 'alcohol')
sns.boxplot(dataset_only[:,11],color="y",ax=axes[3, 2]).set(title = 'quality')
```



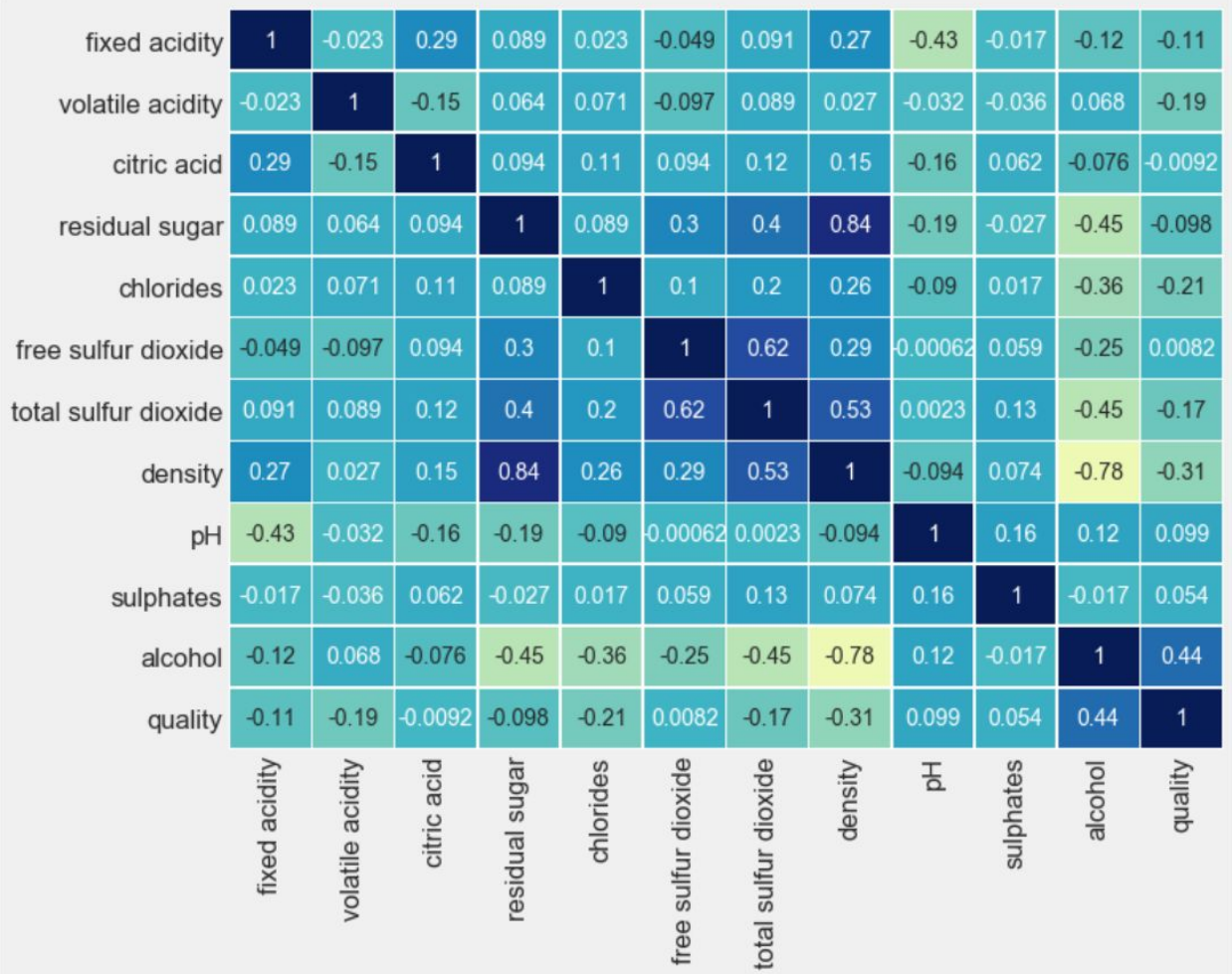
Analysis

Same analysis as our box and whisker plot from Part 1. The quality of more white wine is center around 5-6, putting quality of the wine around 2nd percentage tile. Fixed acidity falls around

25-40% percentile with a lot of out liners from 3-5 and 7-14. Alcohol seems to very fairly contained without many outliers, but most attributes cannot be grouped very cohesively in a box.

4. Heatmap graph (Correlation Matrix Plot)

```
#Correlation plot
corr_matrix = dataset.corr()
pyplot.figure(figsize=(10,8))
sns.heatmap(corr_matrix,annot=True,linewidths=.5,center=0,cbar=False,cmap="YlGnBu")
pyplot.show()
```



Analysis

The plot yields the same results we see from previous work.

The heatmap shows decent correlation between some pairs of data. Positive pearson coefficients indicate a positive correlation, which means as one attribute's value increases, so does the other. Negative pearson coefficients mean that as one attribute's value increases, the other decreases. These correlation pairs may also be indicators of quality, so we can plot these pairs in the next section and observe if this is true.

Notable Positive Correlations (listed from strong to weak):

- [density, residual sugar]
- [free sulfur dioxide, total sulfur dioxide]

Notable Negative Correlations (listed from strong to weak):

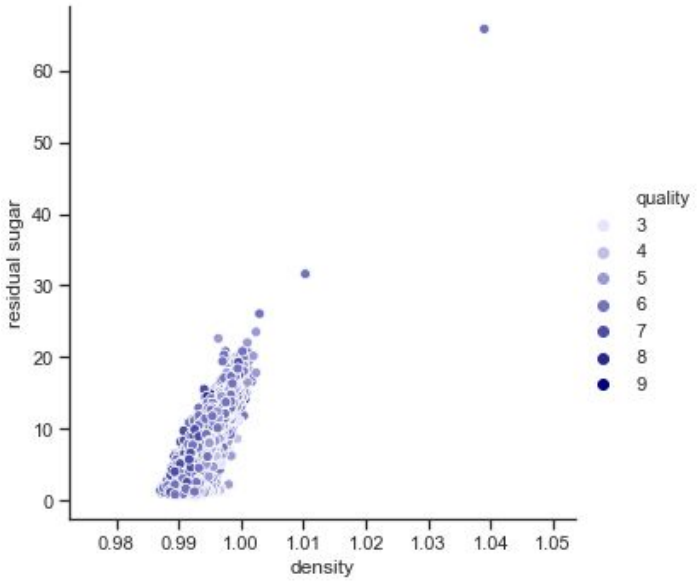
- [density, alcohol]

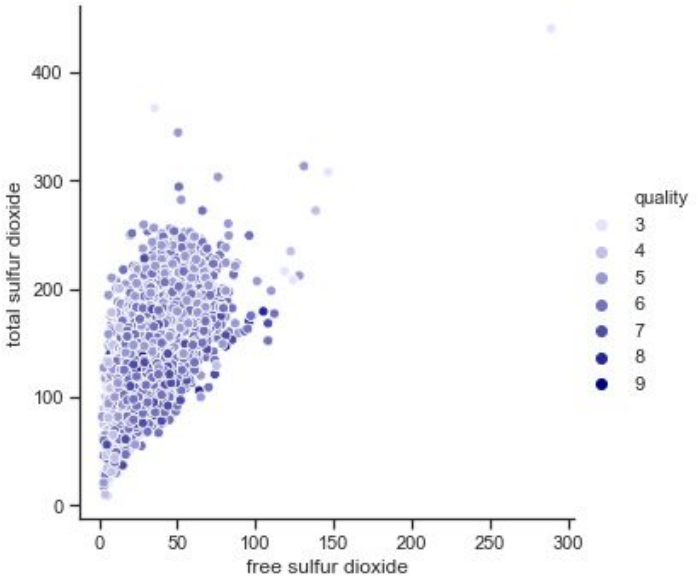
5. Relationship Plots

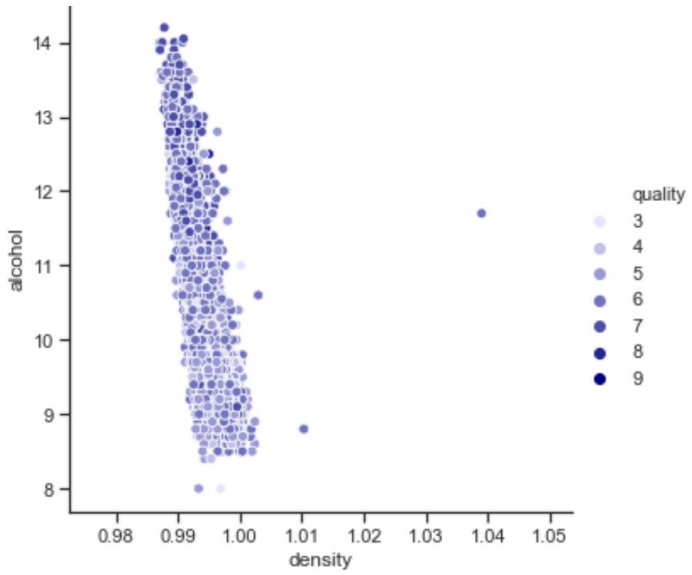
From the correlation plot, we visually observed pairs of data categories that had strong correlations. The correlations were either negative or positive. Using Seaborn relationship plots, we can plot the strongest pairs and observe whether the correlation relates to a change in quality classification for the wine.

```
"""
        #3 Relationship Plots
"""
sns.relplot(x="density", y = "residual sugar", hue = "quality", data=dataset, palette =sns.light_palette("navy",n_colors=7) )
sns.relplot(x="free sulfur dioxide", y = "total sulfur dioxide", hue = "quality", data=dataset, palette =sns.light_palette("navy",n_colors=7))

sns.relplot(x="density", y = "alcohol", hue = "quality", data=dataset, palette =sns.light_palette("navy",n_colors=7))
```







Analysis

We can observe some data category pairs that have a correlation that might be indicative of wine quality. The correlation matrix plot showed that the strongest correlations were:

+[density, residual sugar]

-[density, alcohol]

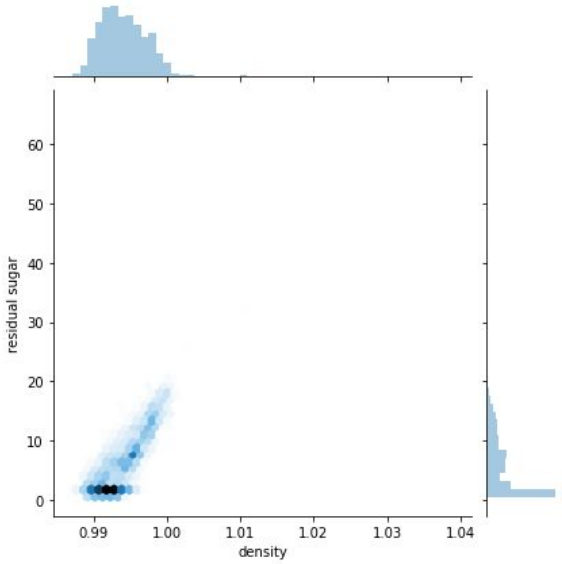
When **density** and **residual sugar** both increase (positive correlation), wine quality seems to slightly decrease. The negative correlation between **density** and **alcohol** seems to show that wine quality is higher when alcohol is high but density is low.

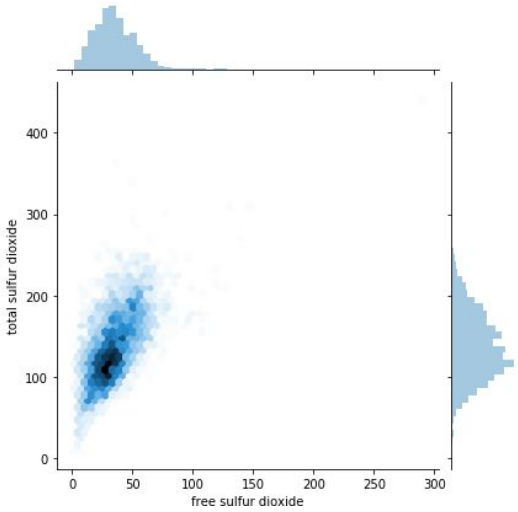
The plots and analysis seem to indicate that these strong correlations also might determine wine quality, so they are good feature pairs to observe and analyze.

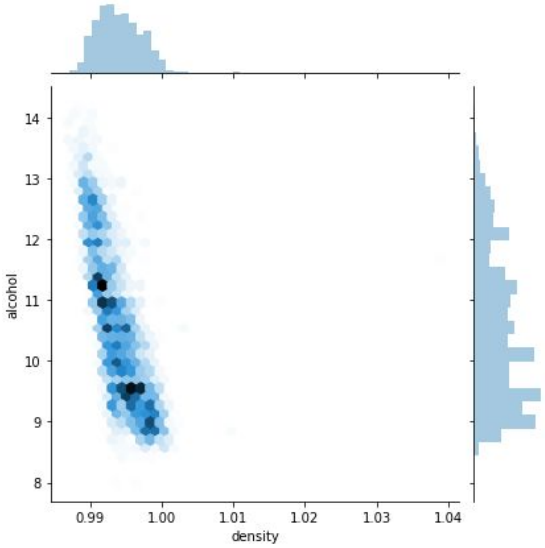
6. Hexbin Plot

We further took these 3 correlation pairs and used hexbin plots to more easily observe the amount of data points in different sections of the plot. This is especially useful for large amounts of data because data is grouped in hex “bins” to more easily discern how dense that area of the plot is with data points.

```
sns.jointplot(x="density", y = "residual sugar",kind = "hex", data = dataset)
sns.jointplot(x="free sulfur dioxide", y = "total sulfur dioxide",kind = "hex", data = dataset)
sns.jointplot(x="alcohol", y = "density",kind = "hex", data = dataset)
```







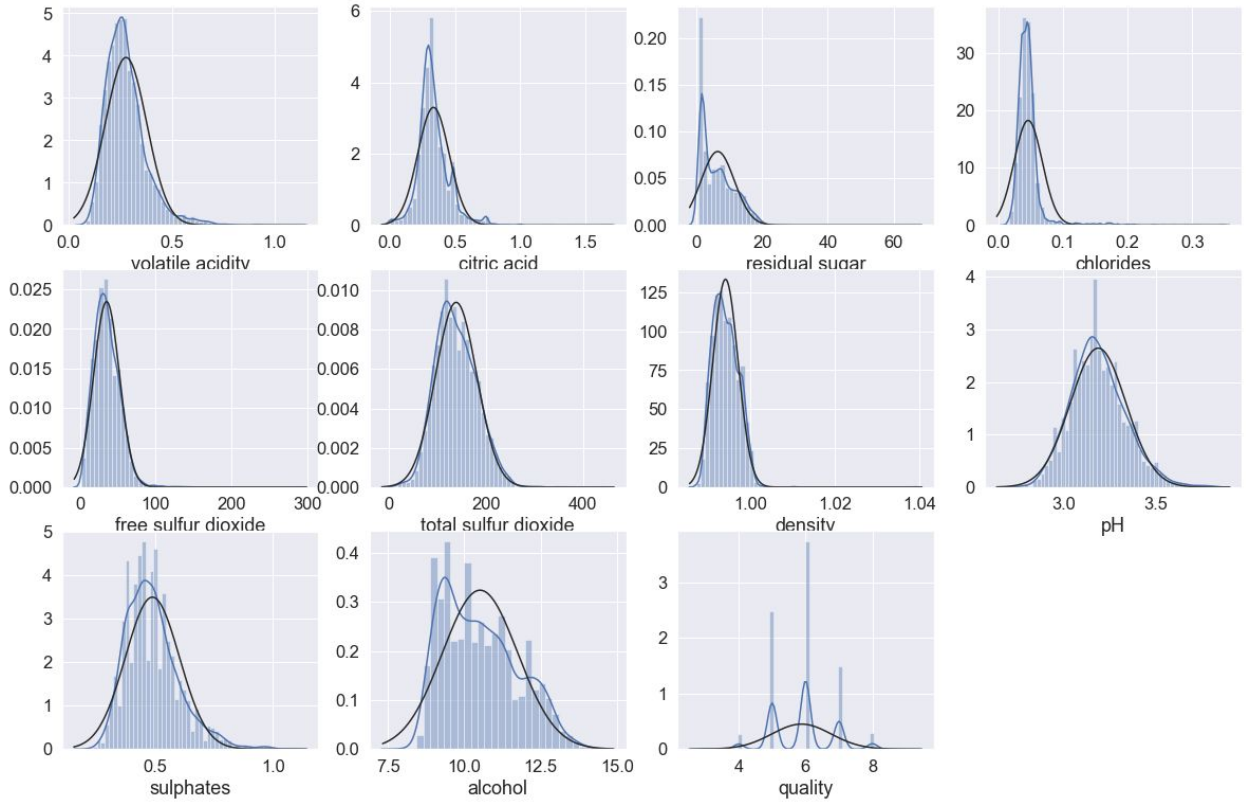
Analysis

The distribution of data is not very uniform in these plots and not much new information is gained. We see a more concentrated amount of data points in the lower left of the total sulfur dioxide & free sulfur dioxide plot, with the concentrations evenly tapering off. Not much can be discerned from these plots, but it is interesting to see the distribution of data based on these pairs.

6. Distribution plot

```
In [26]: from scipy.stats import norm
         pyplot.figure(figsize = (20,22))

         for i in range(1,12):
             pyplot.subplot(5,4,i)
             sns.distplot(dataset[dataset.columns[i]], fit=norm)
```

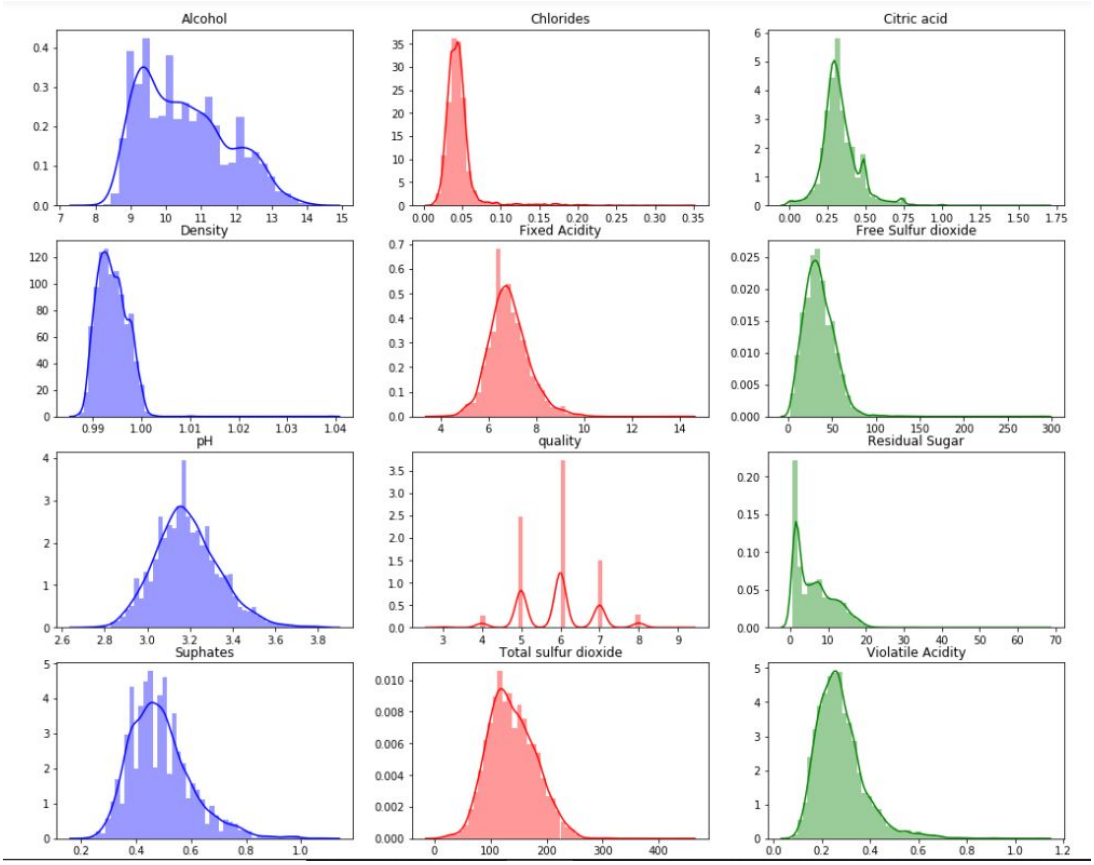


Alternative code and plot:

```
In [82]: # Histograms on each attributes

         f, axes = plt.subplots(4, 3, figsize=(17, 14))

         sns.distplot(dataset_only[:,10], hist=True, color="b", ax=axes[0, 0]).set(title = 'Alcohol')
         sns.distplot(dataset_only[:,4], hist=True, color="r", ax=axes[0, 1]).set(title = 'Chlorides')
         sns.distplot(dataset_only[:,2], hist=True, color="g", ax=axes[0, 2]).set(title = 'Citric acid')
         sns.distplot(dataset_only[:,7], hist=True, color="b", ax=axes[1, 0]).set(title = 'Density')
         sns.distplot(dataset_only[:,0], hist=True, color="r", ax=axes[1, 1]).set(title = 'Fixed Acidity')
         sns.distplot(dataset_only[:,5], hist=True, color="g", ax=axes[1, 2]).set(title = 'Free Sulfur dioxide'
         sns.distplot(dataset_only[:,8], hist=True, color="b", ax=axes[2, 0]).set(title = 'pH')
         sns.distplot(dataset_only[:,11], hist=True, color="r", ax=axes[2, 1]).set(title = 'quality')
         sns.distplot(dataset_only[:,3], hist=True, color="g", ax=axes[2, 2]).set(title = 'Residual Sugar')
         sns.distplot(dataset_only[:,9], hist=True, color="b", ax=axes[3, 0]).set(title = 'Suphates')
         sns.distplot(dataset_only[:,6], hist=True, color="r", ax=axes[3, 1]).set(title = 'Total sulfur dioxide
         sns.distplot(dataset_only[:,1], hist=True, color="g", ax=axes[3, 2]).set(title = 'Violatile Acidity')
```



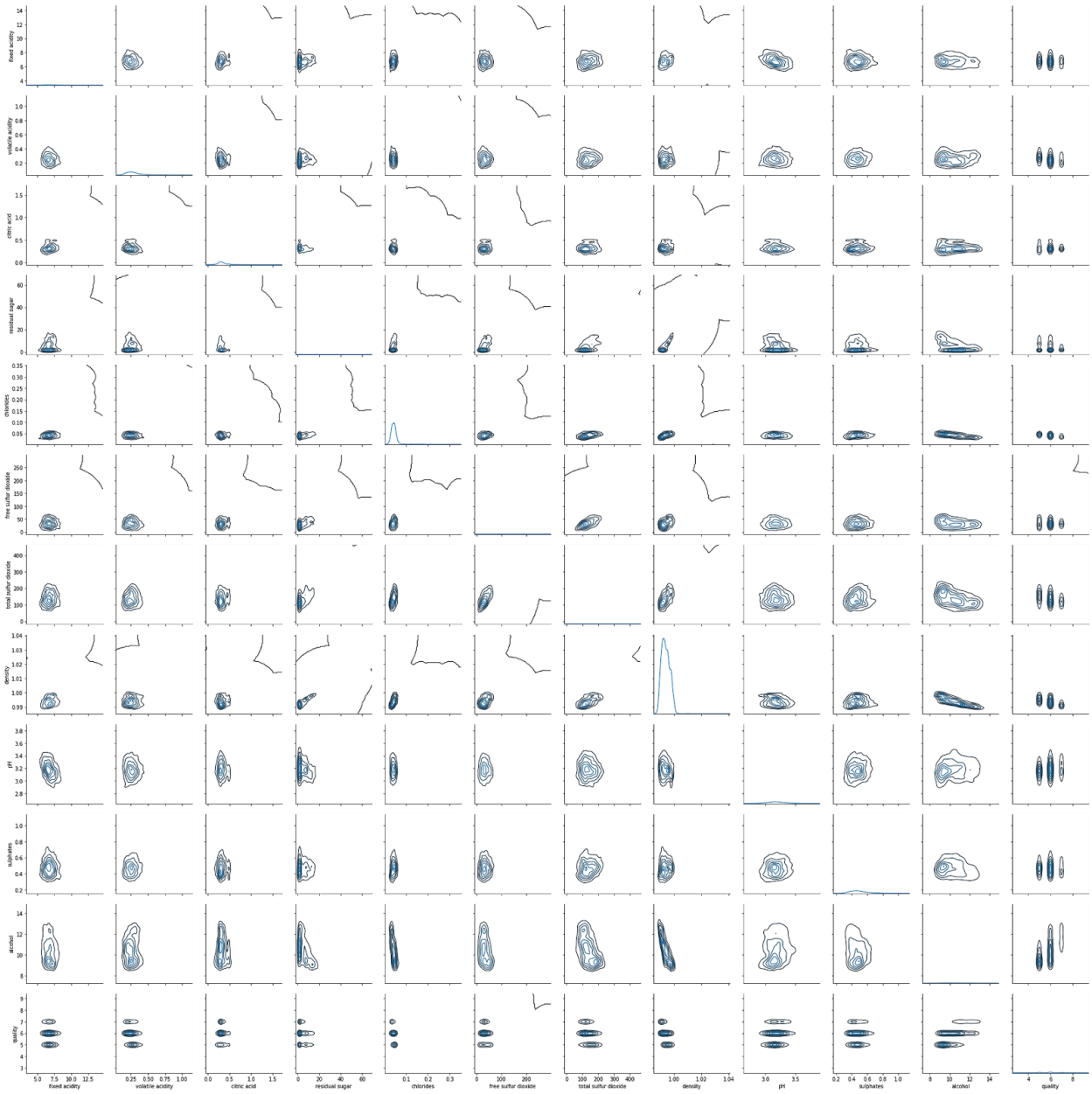
Analysis

We see distributions we saw in earlier parts, but this is just done in Seaborn. We see distributions with both bars and curves and can see where it is more likely to find certain values, as well as observe any possible skews. Most attributes seem to have typical curves, with a few exceptions such as alcohol.

7. Contour plots of attributes

In [30]:
```
# Contour plots of attributes

g = sns.PairGrid(dataset)
g.map_diag(sns.kdeplot)
g.map_offdiag(sns.kdeplot, n_levels=6);
```

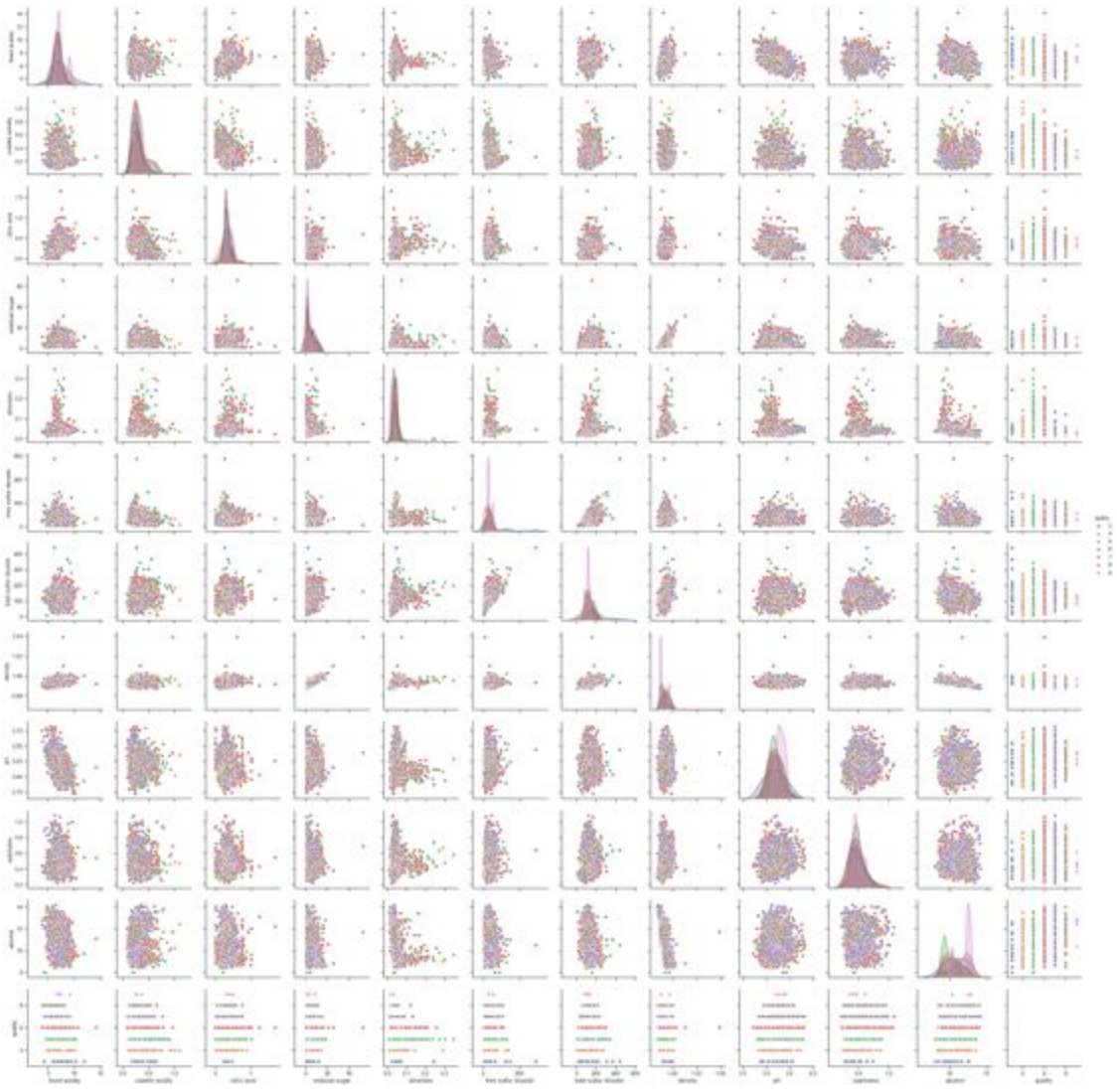


Analysis

This is similar to scatter plots and relationship scatter plots, but there are lines to show contours of the shapes formed in a scatter plot. This can be effective to view groups of data, perhaps differentiated by a classification such as quality. This can be observed in some of these plots, but most of the plots have too much overlapping data so not much can be gained from these contours. It would be more useful on data that is linearly separable and very distinctly separated based on classification.

8. Scatter Pairplot to colorize the quality.

```
"""
        #2 Scatter
"""
sns.set(style = "ticks")
sns_plot = sns.pairplot(dataset, hue = "quality")
sns_plot.savefig("snsScatterPlot.png")
```



Analysis:

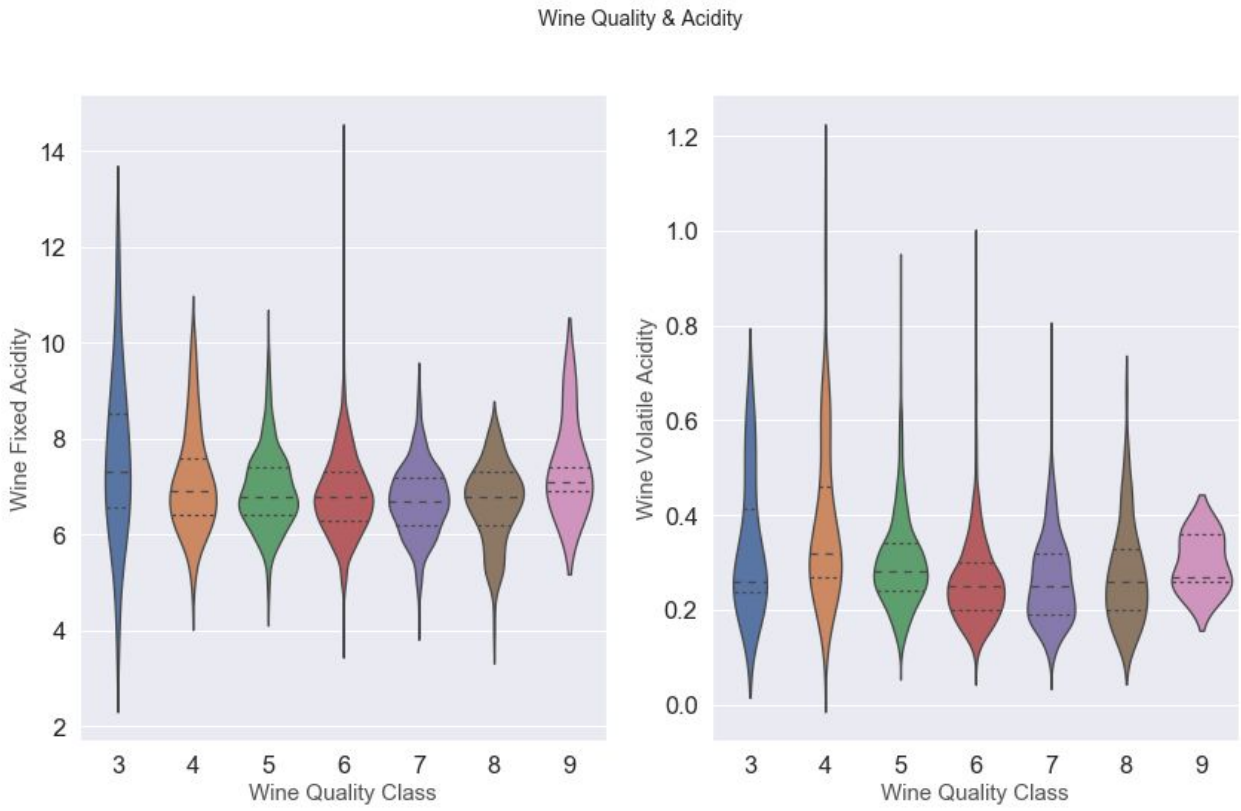
Coloring based on quality in Seaborn allows us to see distribution of quality in plots. However, our data has many classifications that make the plots difficult to extract important information from, as they appear to be a large mess of colors.

9. Violin plot

Quality & Volatile Acidity

```
In [54]: # Quality & Volatile Acidity
f, (ax1, ax2) = pyplot.subplots(1, 2, figsize=(14, 8))
f.suptitle('Wine Quality & Acidity', fontsize=14)
sns.violinplot(x='quality', y='fixed acidity', data=dataset, split=True, inner='quart',
               linewidth=1.3, ax=ax1)
ax1.set_xlabel("Wine Quality Class",size = 15,alpha=0.8)
ax1.set_ylabel("Wine Fixed Acidity",size = 15,alpha=0.8)

sns.violinplot(x='quality', y='volatile acidity', data=dataset, split=True, inner='quart',
               linewidth=1.3, ax=ax2)
ax2.set_xlabel("Wine Quality Class",size = 15,alpha=0.8)
ax2.set_ylabel("Wine Volatile Acidity",size = 15,alpha=0.8)
pyplot.show()
```



Analysis

This is similar to box plots, with another way to simply visualize the data. This allows you to see probability density of the data at different values.

10. Confusion Matrix

Logistic Regression on validation on training data.

In [60]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X = dataset[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides',
             'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']]
y = dataset.quality

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=40)
```

In [63]:
```
# LogisticRegression
lr = LogisticRegression(random_state=40)
lr.fit(X_train, y_train)
```

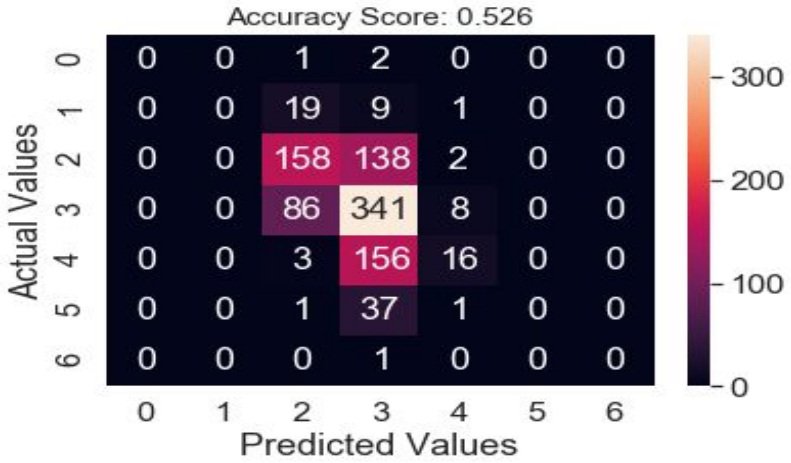
Out[63]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=40, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [64]:
```
train_accuracy = lr.score(X_train, y_train)
test_accuracy = lr.score(X_test, y_test)
print('Accuracy in Train Group   : {:.2f}'.format(train_accuracy),
      'Accuracy in Test  Group   : {:.2f}'.format(test_accuracy), sep='\n')
```

```
Accuracy in Train Group   : 0.53
Accuracy in Test  Group   : 0.53
```

In [65]:
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix as cm

predictions = lr.predict(X_test)
score = round(accuracy_score(y_test, predictions), 3)
cm1 = cm(y_test, predictions)
sns.heatmap(cm1, annot=True, fmt=".0f")
pyplot.xlabel('Predicted Values')
pyplot.ylabel('Actual Values')
pyplot.title('Accuracy Score: {0}'.format(score), size = 15)
pyplot.show()
```



Analysis

There are a lot of misclassifications and you can see the accuracy is quite low at about 52.6%. This is also similar to when we did this matrix in Part 1, which had almost the same accuracy percentage for Logistic Regression.

11. Count Plot on Cross Validation

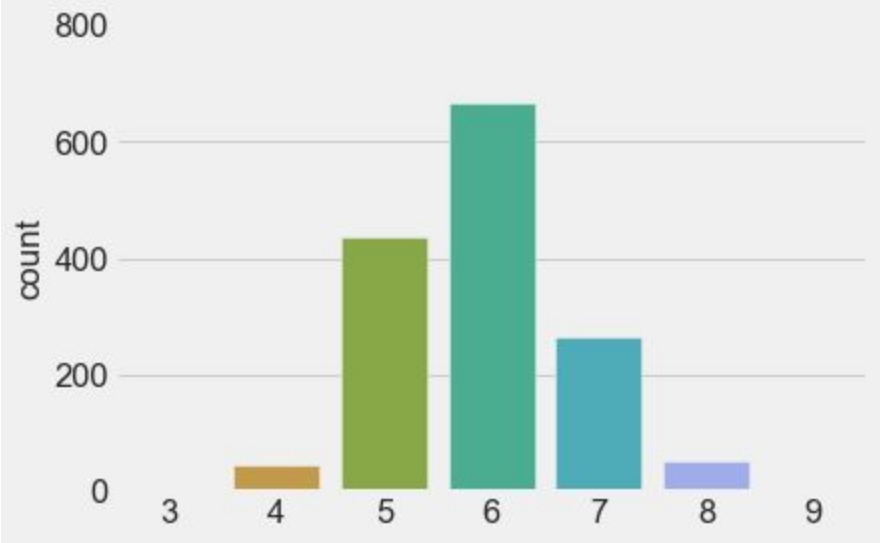
In [66]:
```
# Cross Validation
X = dataset.drop(['quality'], axis=1)
y = dataset.quality
y = np.array(y)
```

In [67]:
```
pyplot.style.use('fivethirtyeight')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=40)
print("Number of Rows in    Training dataset :  {} ".format(len(X_train)))
print("Number of Targets in Training dataset :  {} ".format(len(y_train)))
print("Number of Rows in    Test dataset :  {} ".format(len(X_test)))
print("Number of Targets in Test dataset :  {} ".format(len(y_test)))
```

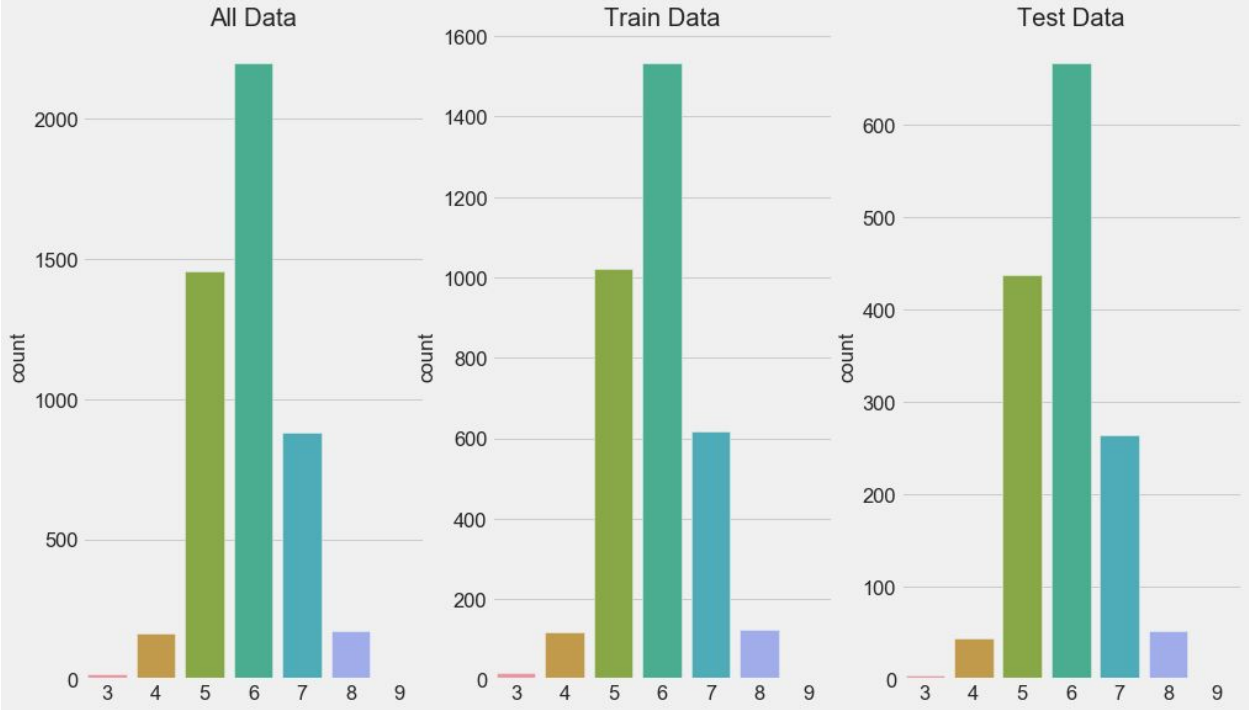
```
Number of Rows in    Training dataset :  3428
Number of Targets in Training dataset :  3428
Number of Rows in    Test dataset :  1470
Number of Targets in Test dataset :  1470
```

In [71]:
```
sns.countplot(y_test)
pyplot.ylim((0,800))
```



In [72]:
```
pyplot.figure(figsize=(15,9))
y_list = [y, y_train, y_test]
titles = ['All Data','Train Data', 'Test Data']

for i in range(1,4):
    pyplot.subplot(1,3,i)
    sns.countplot(y_list[i-1])
    pyplot.title(titles[i-1])
```



Analysis

This creates training and test data with similar distributions of the whole dataset, for use with machine learning.