Wine Quality Prediction

By Hakeem Lawrence

Dataset Link: https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
```

In [2]:
```
#importing dataset and viewing first 5 rows

wine = pd.read_csv('winequality-red.csv')
wine.head(10)
```

Out[2]:

	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
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5

In [3]:
```
#viewing number of rows & columns

wine.shape
```

Out[3]:
```
(1599, 12)
```

In [4]:
```
#viewing number of missing values

wine.isna().sum()
```

Out[4]:
```
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
```

Descriptive Stats

In [5]:
```
wine.describe().T.round(3)
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.320	1.741	4.600	7.100	7.900	9.200	15.900
volatile acidity	1599.0	0.528	0.179	0.120	0.390	0.520	0.640	1.580
citric acid	1599.0	0.271	0.195	0.000	0.090	0.260	0.420	1.000
residual sugar	1599.0	2.539	1.410	0.900	1.900	2.200	2.600	15.500
chlorides	1599.0	0.087	0.047	0.012	0.070	0.079	0.090	0.611
free sulfur dioxide	1599.0	15.875	10.460	1.000	7.000	14.000	21.000	72.000
total sulfur dioxide	1599.0	46.468	32.895	6.000	22.000	38.000	62.000	289.000
density	1599.0	0.997	0.002	0.990	0.996	0.997	0.998	1.004
pH	1599.0	3.311	0.154	2.740	3.210	3.310	3.400	4.010
sulphates	1599.0	0.658	0.170	0.330	0.550	0.620	0.730	2.000
alcohol	1599.0	10.423	1.066	8.400	9.500	10.200	11.100	14.900
quality	1599.0	5.636	0.808	3.000	5.000	6.000	6.000	8.000

In [6]:
```
#Counting all values from the output variable

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

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

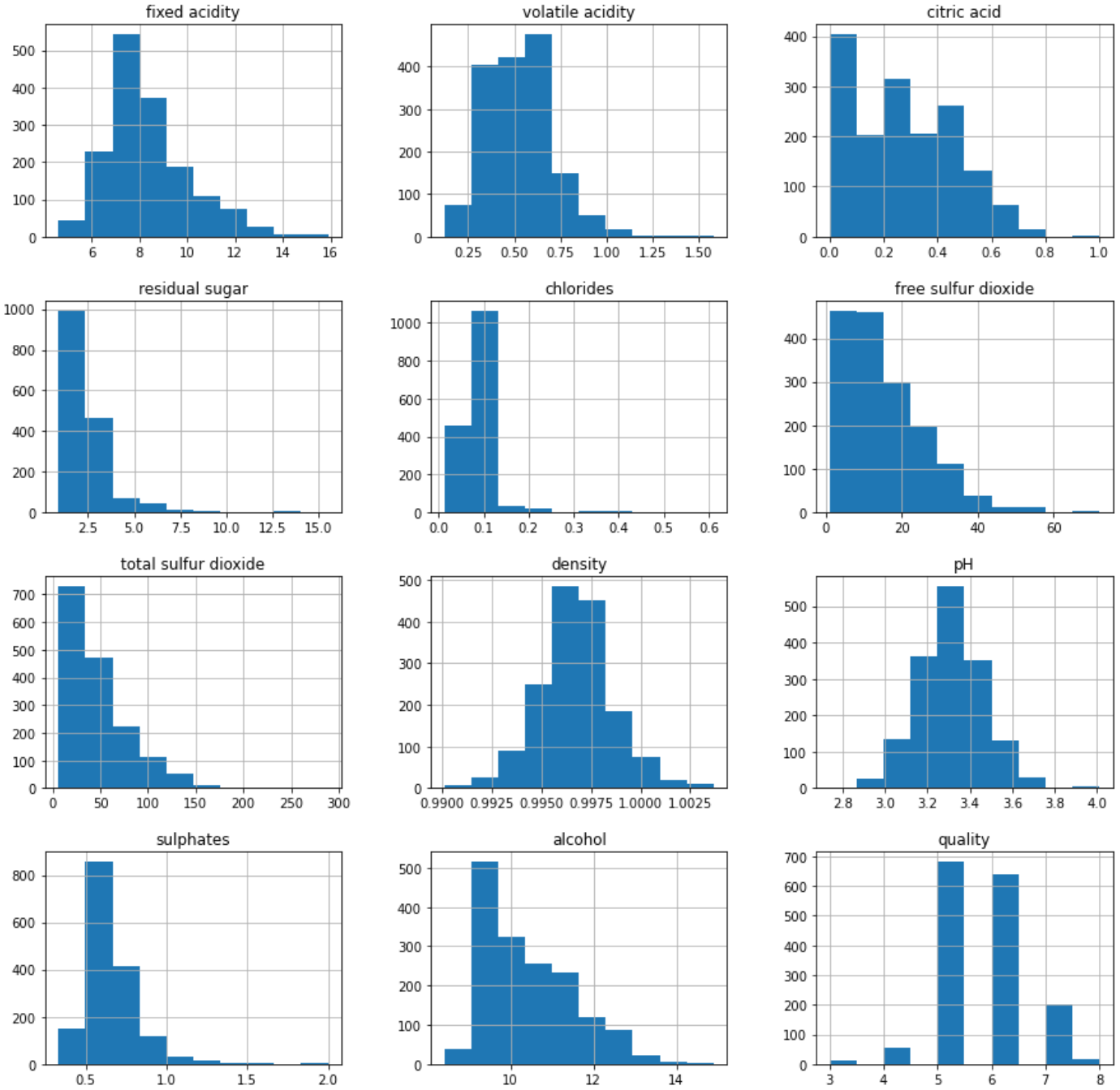
Viewing distribution for each variable

In [7]:
```
#viewig the distribution of all variables

wine.hist(figsize= (15,15))
```

Out[7]:
```
array([[<AxesSubplot:title={'center':'fixed acidity'}>,
        <AxesSubplot:title={'center':'volatile acidity'}>,
        <AxesSubplot:title={'center':'citric acid'}>],
       [<AxesSubplot:title={'center':'residual sugar'}>,
        <AxesSubplot:title={'center':'chlorides'}>,
        <AxesSubplot:title={'center':'free sulfur dioxide'}>],
       [<AxesSubplot:title={'center':'total sulfur dioxide'}>,
        <AxesSubplot:title={'center':'density'}>,
```

```
 <AxesSubplot:title={'center':'pH'}>],
[<AxesSubplot:title={'center':'sulphates'}>,
 <AxesSubplot:title={'center':'alcohol'}>,
 <AxesSubplot:title={'center':'quality'}>]], dtype=object)
```



Examining the relationship between each input and output variable

In [8]:

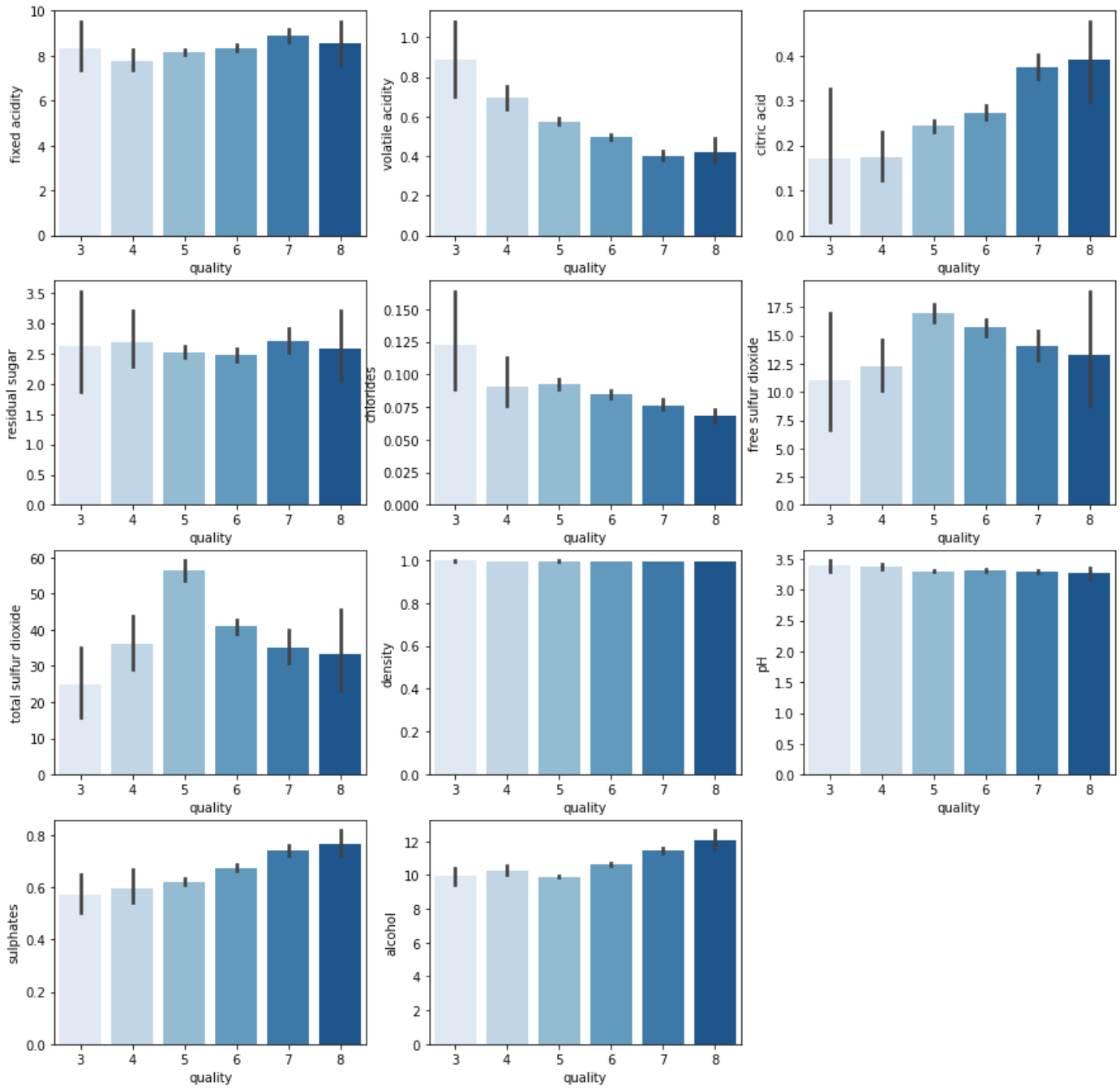
```
#viewing the relationship between inputs and output variable

fig, axes = plt.subplots(4,3, figsize=(15,15))

# plot = plt.figure(figsize =(5,5))
sns.barplot(data=wine,x='quality', y="fixed acidity", palette='Blues',ax=axes[0,0])
sns.barplot(data=wine,x='quality', y="volatile acidity", palette='Blues',ax=axes[0,1])
sns.barplot(data=wine,x='quality', y="citric acid", palette='Blues',ax=axes[0,2])
sns.barplot(data=wine,x='quality', y="residual sugar", palette='Blues',ax=axes[1,0])
sns.barplot(data=wine,x='quality', y="chlorides", palette='Blues',ax=axes[1,1])
sns.barplot(data=wine,x='quality', y="free sulfur dioxide", palette='Blues',ax=axes[1,2]
sns.barplot(data=wine,x='quality', y="total sulfur dioxide",palette='Blues', ax=axes[2,0
sns.barplot(data=wine,x='quality', y="density",palette='Blues', ax=axes[2,1])
sns.barplot(data=wine, x='quality',y="pH", palette='Blues', ax=axes[2,2] )
sns.barplot(data=wine,x='quality', y="sulphates",palette='Blues', ax=axes[3,0] )
```

```
sns.barplot(data=wine,x='quality', y="alcohol",palette='Blues', ax=axes[3,1] )
fig.delaxes(axes[3,2])


plt.show()
```



Finding the Correlation between input and output variable

In [9]:
```
#Finding correlation between variables

corr = wine.corr()
```

In [10]:
```
#viewing column names

wine.columns
```

Out[10]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [11]:
```
#Finding correlation between all inputs and output variable

wine.corr()[['quality']]

cm= sns.color_palette("light:b", as_cmap=True)
corr_analysis = pd.DataFrame(wine[['fixed acidity', 'volatile acidity', 'citric acid', '
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH','sulphates', 'alcohol', 'quality']]).corr()['quality'].reset_index()

corr_analysis.columns=['input variables', 'correlation']
corr_analysis = corr_analysis[corr_analysis['input variables']!= 'quality']
corr_analysis = corr_analysis.sort_values('correlation', ascending =False)
corr_analysis.style.background_gradient(cmap=cm).set_precision(2)
```

```
C:\Users\hakee\AppData\Local\Temp\ipykernel_19252\3206807291.py:13: FutureWarning: this
method is deprecated in favour of `Styler.format(precision=..)`
  corr_analysis.style.background_gradient(cmap=cm).set_precision(2)
```

Out[11]:

	input variables	correlation
10	alcohol	0.48
9	sulphates	0.25
2	citric acid	0.23
0	fixed acidity	0.12
3	residual sugar	0.01
5	free sulfur dioxide	-0.05
8	pH	-0.06
4	chlorides	-0.13
7	density	-0.17
6	total sulfur dioxide	-0.19
1	volatile acidity	-0.39

Pre-Processing

In [12]:
```
#Splitting input from output variable

X =  wine.drop(columns='quality', axis=1)
```

Label Binarization

1= Good Quality 0= Bad Quality

In [13]:
```
#Grouping output values into binary classes

Y= wine['quality'].apply(lambda y_value: 1 if y_value>=6 else 0)
print(Y.value_counts())
```

```
1    855
0    744
Name: quality, dtype: int64
```

In [14]:
```
#showing output of value proportions

((Y.value_counts()/len(wine))*100).round(2)
```

Out[14]:
```
1    53.47
```

```
0    46.53
Name: quality, dtype: float64
```

Training the model

In [15]:
```
#creating/splitting the training and testing data

X_train, X_test,Y_train, Y_test = train_test_split(X, Y, test_size=.33, random_state= 42
```

In [16]:
```
#examining the number of rows and columns in original, training & testing data

print(X.shape, X_train.shape, X_test.shape)
```

```
(1599, 11) (1071, 11) (528, 11)
```

Random Forest Model

In [17]:
```
#training the data

model = RandomForestClassifier()
model.fit(X_train, Y_train)
```

Out[17]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

Evaluating the Model

Getting Accuracy Score

In [18]:
```
X_test_predict = model.predict(X_test)
testing_accuracy = accuracy_score(X_test_predict, Y_test)

print('Accuracy Score:\n')
print('Testing Accuracy:', testing_accuracy)
```

```
Accuracy Score:

Testing Accuracy: 0.8068181818181818
```

Testing the predictive system

In [19]:
```
#testing with values from the 8th row
input_data = (7.3,0.65,0.0,1.2,0.065,15.0,21.0,0.9946,3.39,0.47,10.0)

#changing input data to numpy array
input_data_np = np.asarray(input_data)

#reshaping the data
input_data_reshaped = input_data_np.reshape(1,-1)

#making prediction
prediction = model.predict(input_data_reshaped)


print(prediction)
if (prediction[0]==1):
    print('The wine is good quality')
else:
    print('The wine is bad quality')
```

```
[1]
The wine is good quality
C:\Users\hakee\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not
have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```