Wine_Quality

March 3, 2022

1 Importing necessary libraries

[72]:
```
import pandas as pd
import numpy as np
import scipy as sci
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
sns.set()
style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

1.0.1 Loading Data Set

[73]:
```
wine_data = pd.read_csv("winequality-red.csv",sep=";")
```

2 Data Information

[74]:
```
wine_data.shape
```

[74]:
```
(1599, 12)
```

2.0.1 Printing First 5 columns to look at data

[75]:
```
wine_data.head()
```

[75]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
```

```
1            7.8              0.88         0.00             2.6       0.098
2            7.8              0.76         0.04             2.3       0.092
3           11.2              0.28         0.56             1.9       0.075
4            7.4              0.70         0.00             1.9       0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

2.0.2 Chacking for Null values

[76]:
```
wine_data.isna().sum()
```

[76]:
```
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
```

3 Data Preprocessing

3.1 Descriptive statistics

[77]:
```
wine_data.describe().T
```

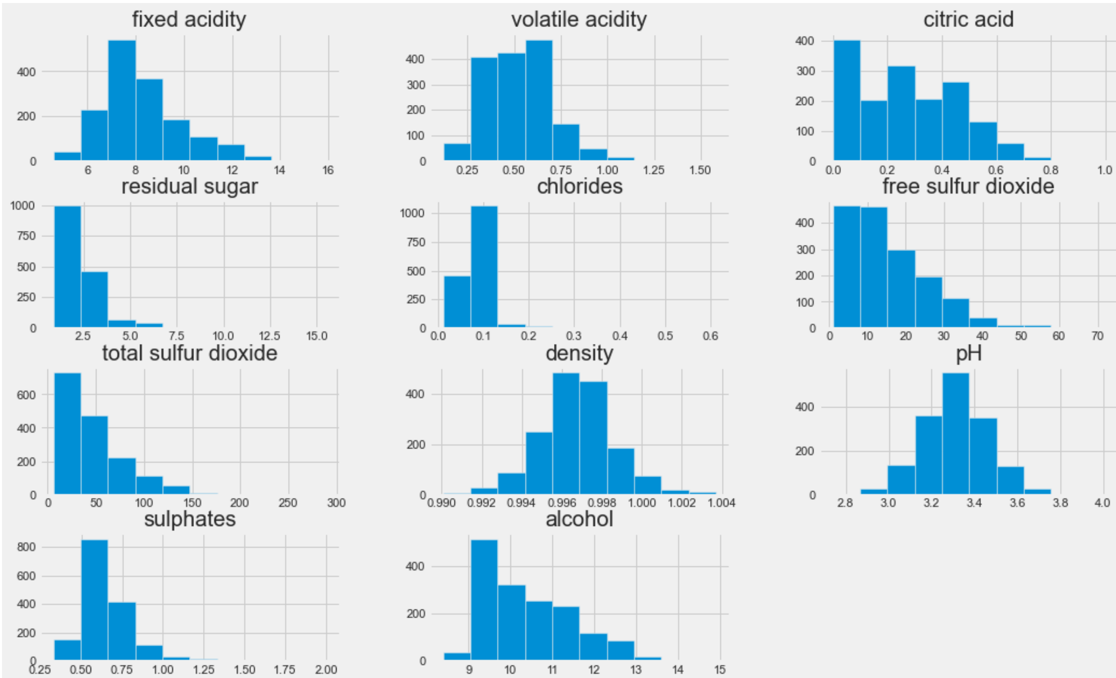
[77]:
```
                      count      mean       std      min     25%  \
fixed acidity        1599.0  8.319637  1.741096  4.60000  7.1000
volatile acidity     1599.0  0.527821  0.179060  0.12000  0.3900
citric acid          1599.0  0.270976  0.194801  0.00000  0.0900
```

```
residual sugar        1599.0   2.538806   1.409928  0.90000   1.9000
chlorides             1599.0   0.087467   0.047065  0.01200   0.0700
free sulfur dioxide   1599.0  15.874922  10.460157  1.00000   7.0000
total sulfur dioxide  1599.0  46.467792  32.895324  6.00000  22.0000
density               1599.0   0.996747   0.001887  0.99007   0.9956
pH                    1599.0   3.311113   0.154386  2.74000   3.2100
sulphates             1599.0   0.658149   0.169507  0.33000   0.5500
alcohol               1599.0  10.422983   1.065668  8.40000   9.5000
quality               1599.0   5.636023   0.807569  3.00000   5.0000

                           50%        75%        max
fixed acidity          7.90000   9.200000   15.90000
volatile acidity       0.52000   0.640000    1.58000
citric acid            0.26000   0.420000    1.00000
residual sugar         2.20000   2.600000   15.50000
chlorides              0.07900   0.090000    0.61100
free sulfur dioxide   14.00000  21.000000   72.00000
total sulfur dioxide  38.00000  62.000000  289.00000
density                0.99675   0.997835    1.00369
pH                     3.31000   3.400000    4.01000
sulphates              0.62000   0.730000    2.00000
alcohol               10.20000  11.100000   14.90000
quality                6.00000   6.000000    8.00000
```

3.2 Exploratory Data Analysis

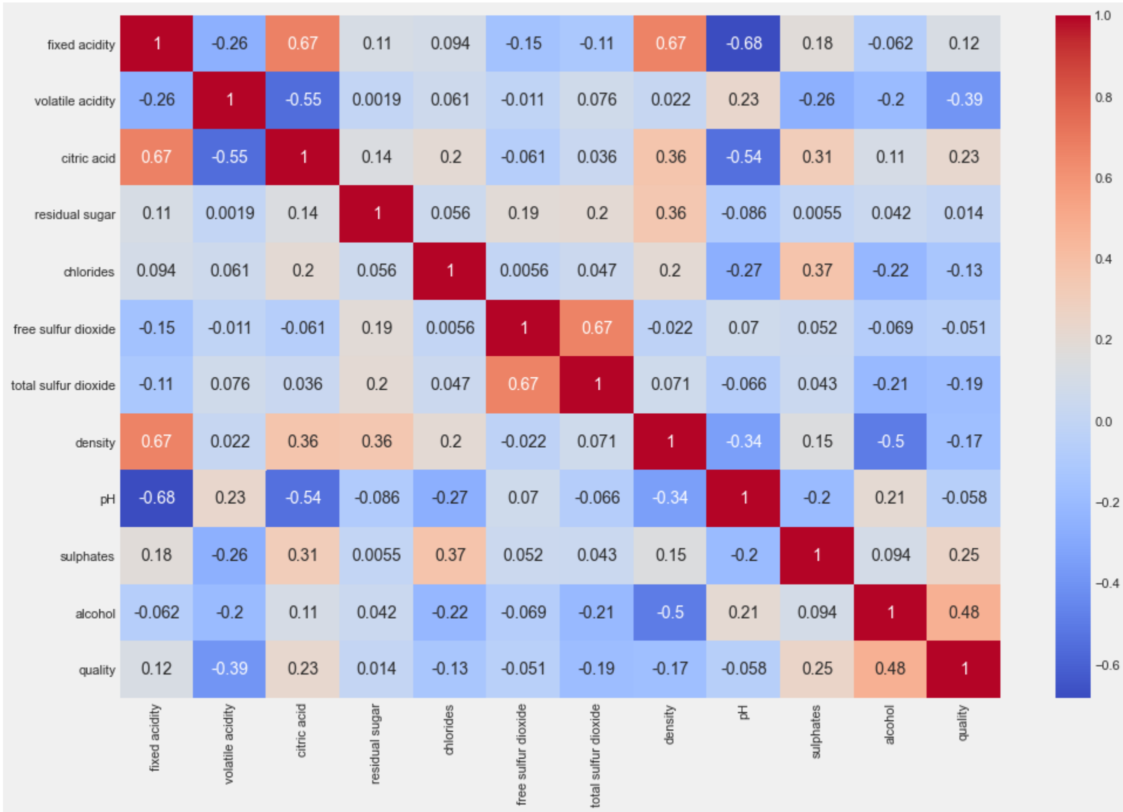
3.2.1 Plotting histogram for each feature for better understanding

[78]:
```python
attribute_only = wine_data.drop(['quality'], axis=1)
plt.rcParams["figure.figsize"] = (16,10)
attribute_only.hist()
plt.show()
```



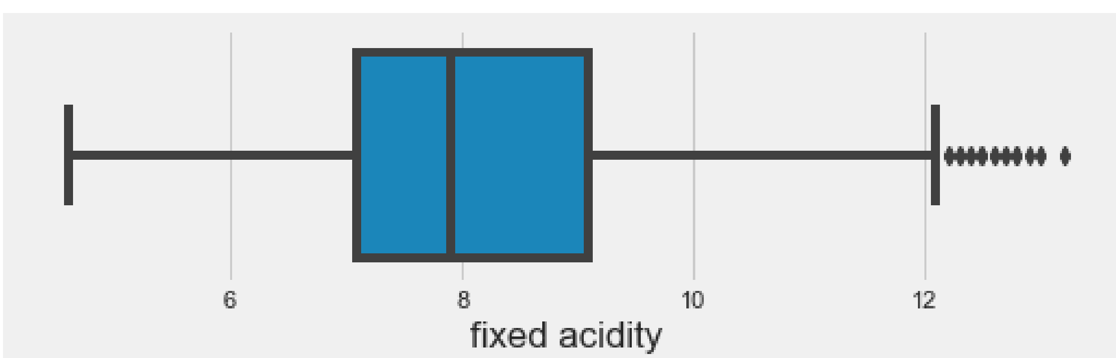
3.2.2 Checking correlation between columns

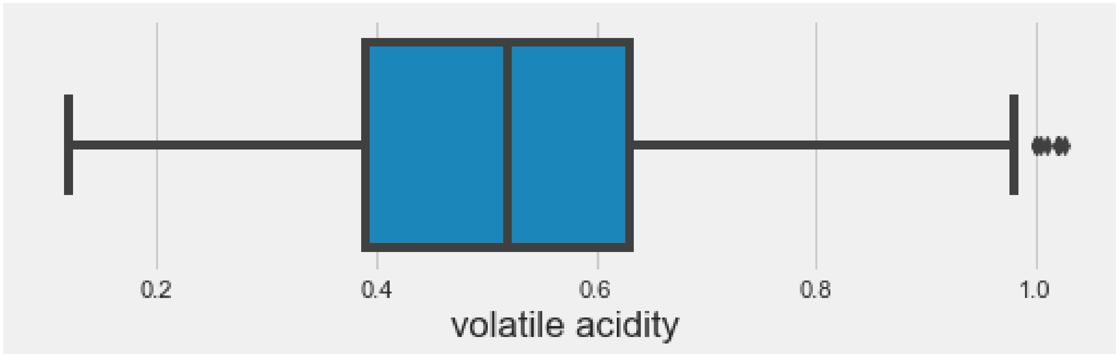
[8]:
```
plt.rcParams["figure.figsize"] = (15,10)
sns.heatmap(wine_data.corr(), annot = True, cmap = 'coolwarm')
plt.show()
```

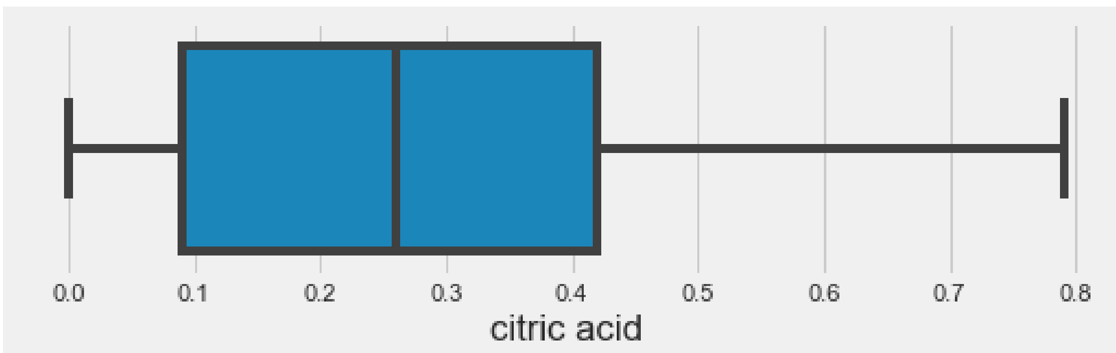


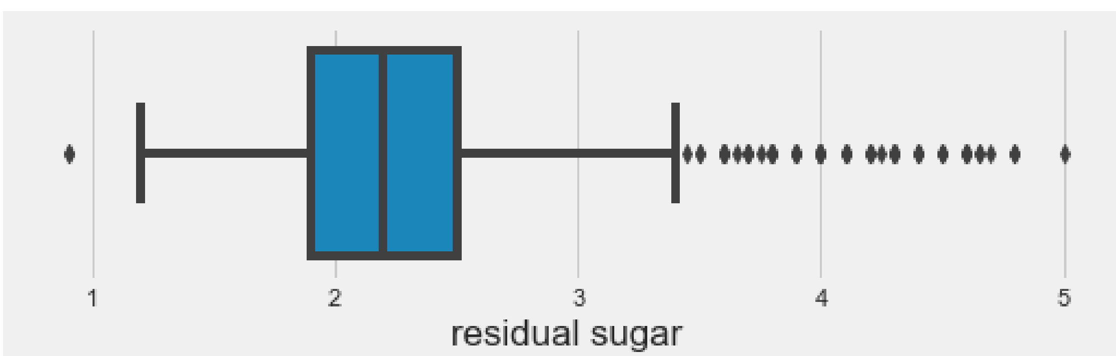
[24]:
```
plt.rcParams["figure.figsize"] = (8,2)
```

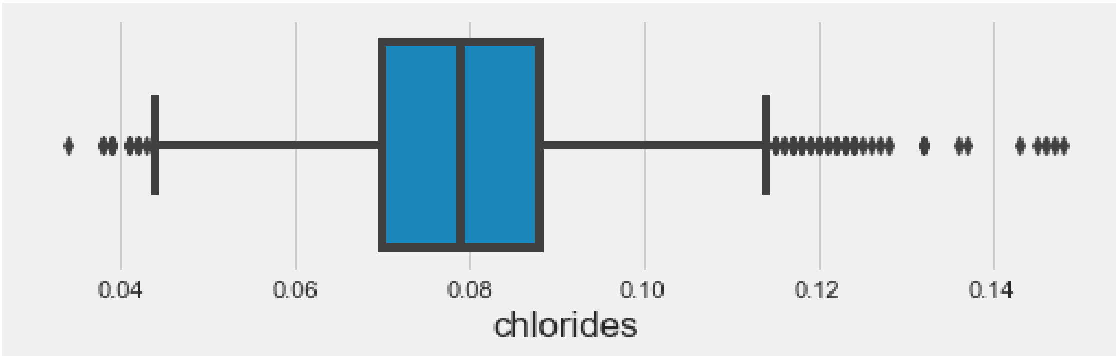
[46]:
```
# Plotting each feature to check outliers
Column_names =['fixed acidity', 'volatile acidity', 'citric acid', 'residual␣
↪sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol']
for col in Column_names:
    sns.boxplot(x=col,data=wine_data)
    plt.show()
```

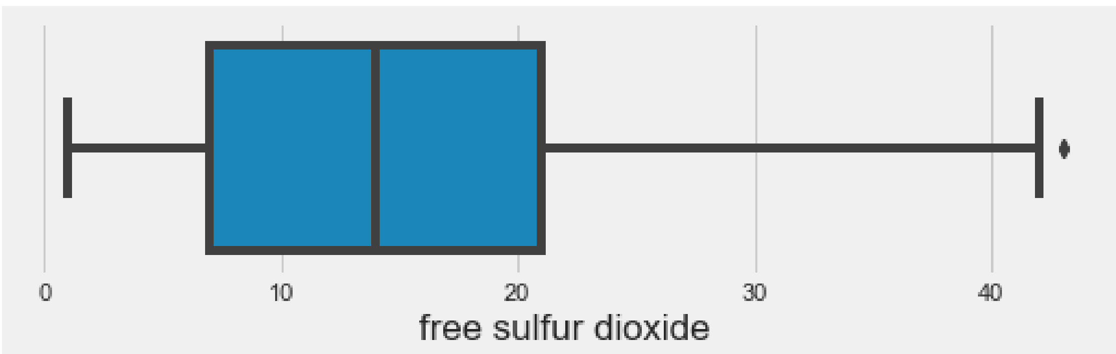


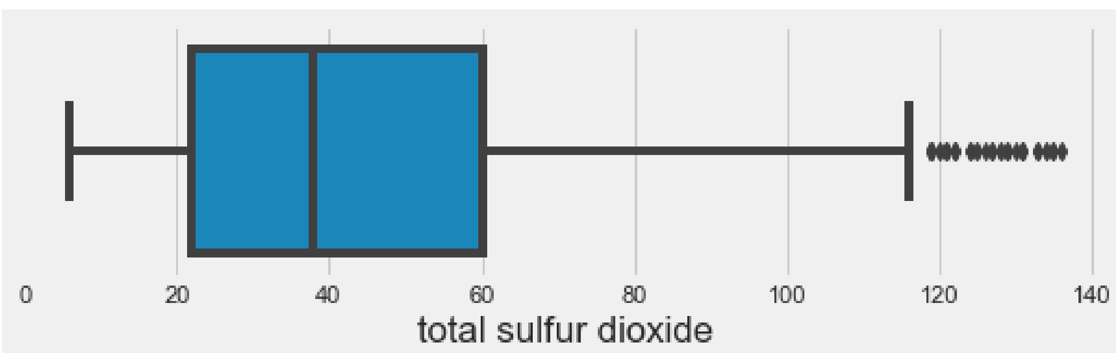

0.2
0.4
0.6
0.8
1.0
volatile acidity

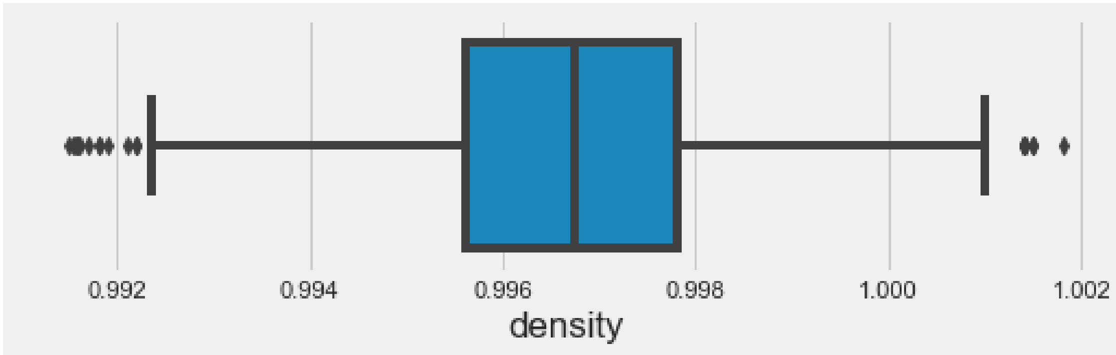

0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
citric acid

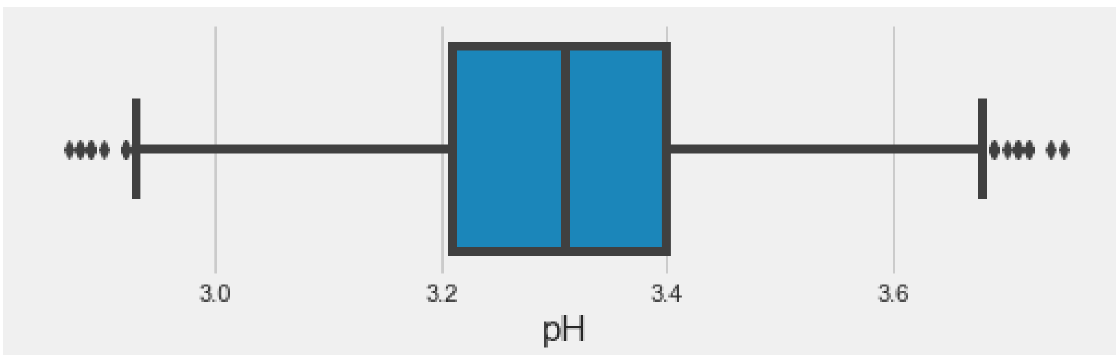

1
2
3
4
5
residual sugar

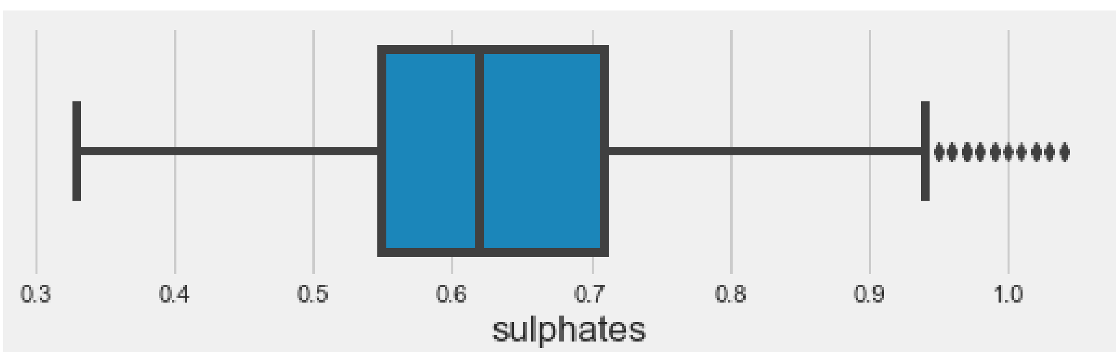

0.04
0.06
0.08
0.10
0.12
0.14
chlorides

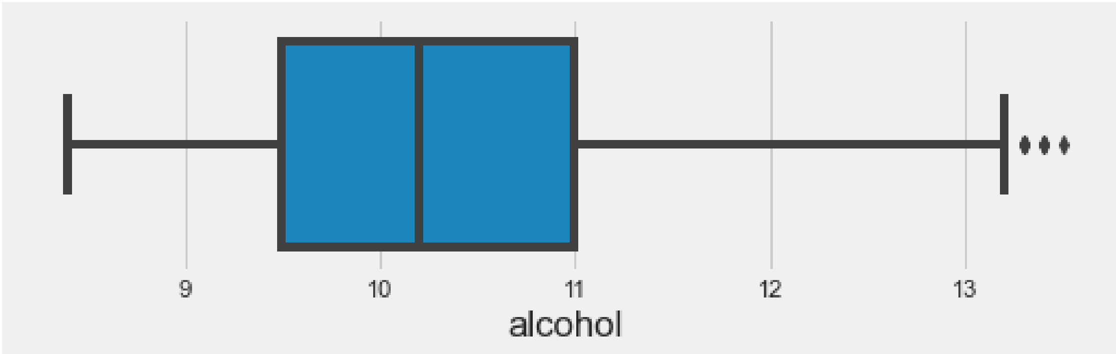

0
10
20
30
40
free sulfur dioxide


0
20
40
60
80
100
120
140
total sulfur dioxide


0.992
0.994
0.996
0.998
1.000
1.002
density


3.0
3.2
3.4
3.6
pH


0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
sulphates

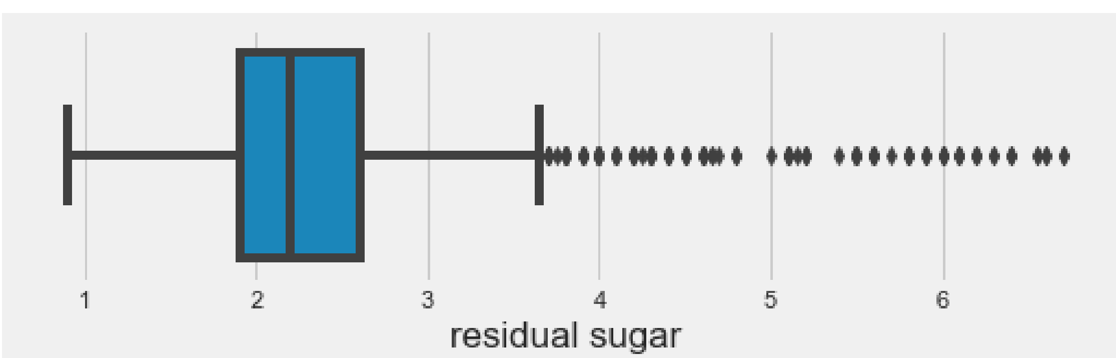


[47]:
```
# Writing a function to remove outliers

def outlier_removed (c):

    a = wine_data[c].mean() + 3*wine_data[c].std()
    b = wine_data[c].mean() - 3*wine_data[c].std()
    wine_data[c] = np.where(
    wine_data[c]>a,
    wine_data[c].mean(),
    np.where(
        wine_data[c]< b,
        wine_data[c].mean(),
        wine_data[c]
    )
)
```

[44]:
```
#Removing outliers
for col in Column_names:
    outlier_removed(col)
```

[43]:
```
#Checking boxplot after removing outlier
sns.boxplot(x='residual sugar',data=wine_data)
```

[43]: <AxesSubplot:xlabel='residual sugar'>



3.2.3 Looking at unique value for Dependent variables

[29]:
```
wine_data['quality'].value_counts()
```

[29]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

3.2.4 Mapping Quality to three classes (0 = Low quality, 1 = Medium quality and 2 = Good quality)

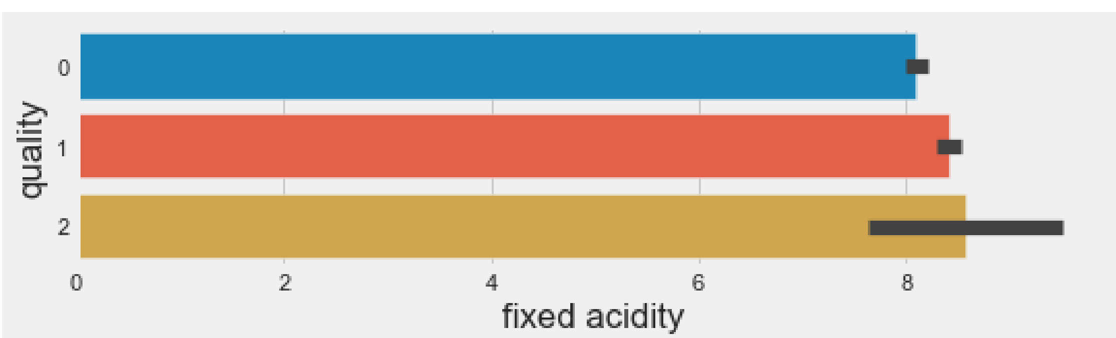
[30]:
```
bins_size = [0, 5.5, 7.5, 10]
label = [0, 1, 2]
wine_data['quality'] = pd.cut(wine_data['quality'], bins=bins_size,␣
 ↪labels=label)
```

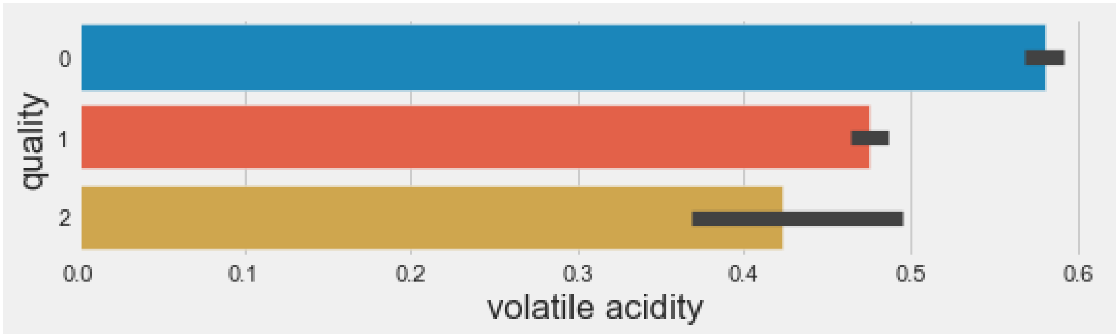
[31]:
```
wine_data['quality'].value_counts()
```

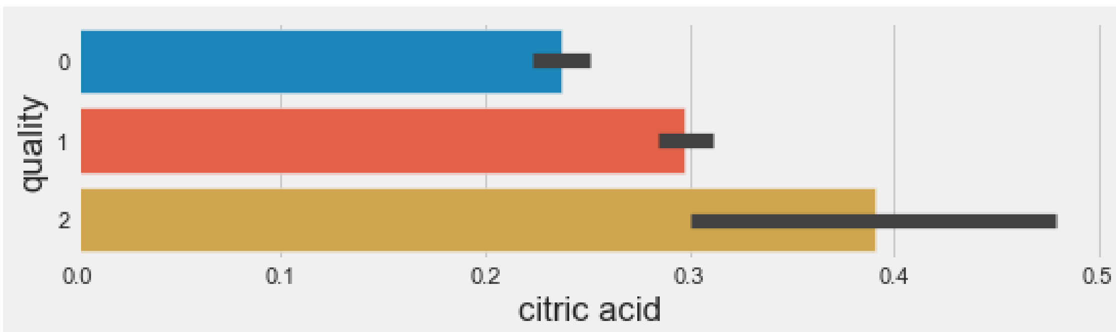
[31]:
```
1    837
0    744
2     18
Name: quality, dtype: int64
```

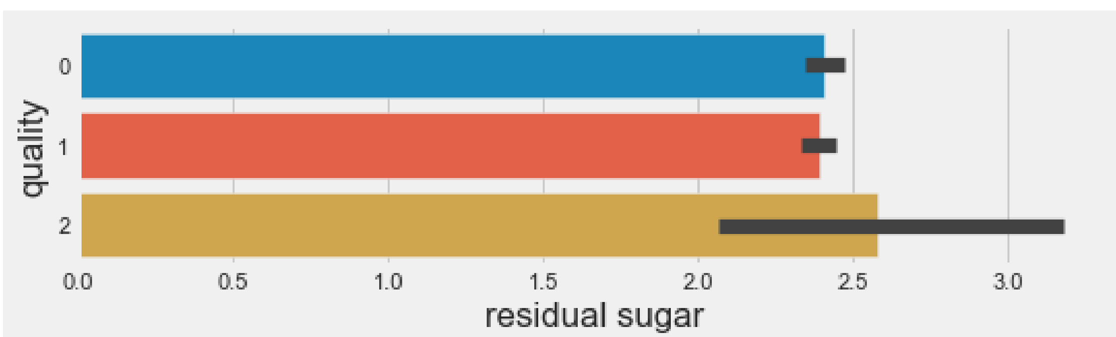
3.2.5 Checking relationship of features with Quality

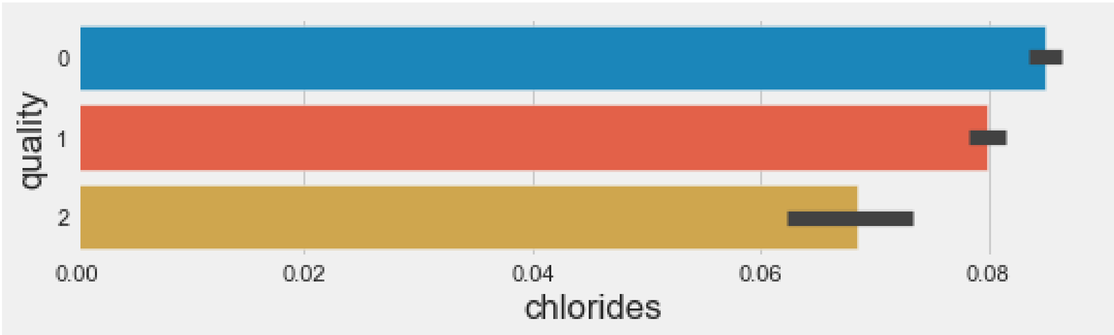
[38]:
```
plt.rcParams["figure.figsize"] = (8,2)
for col in Column_names:
    sns.barplot(x=col,y = 'quality', data=wine_data)
    plt.show()
```

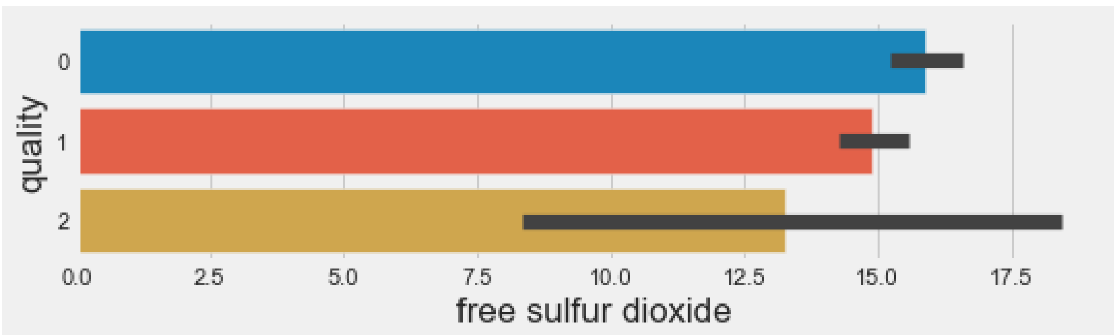


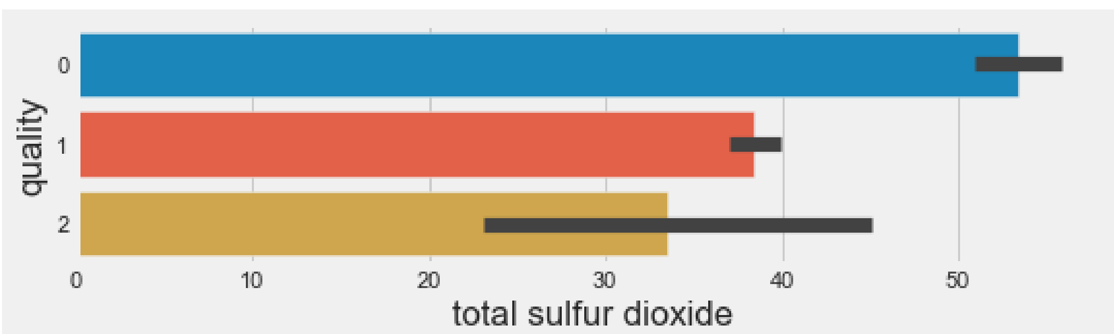

0
1
2
quality
0.0
0.1
0.2
0.3
0.4
0.5
0.6
volatile acidity

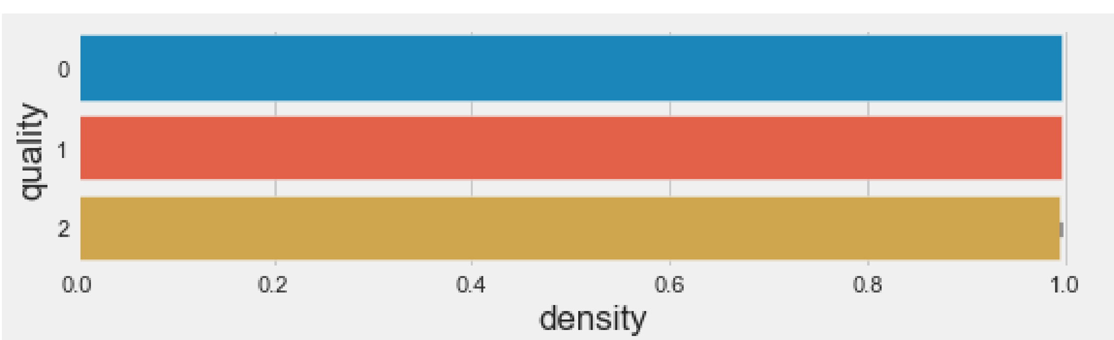

0
1
2
quality
0.0
0.1
0.2
0.3
0.4
0.5
citric acid

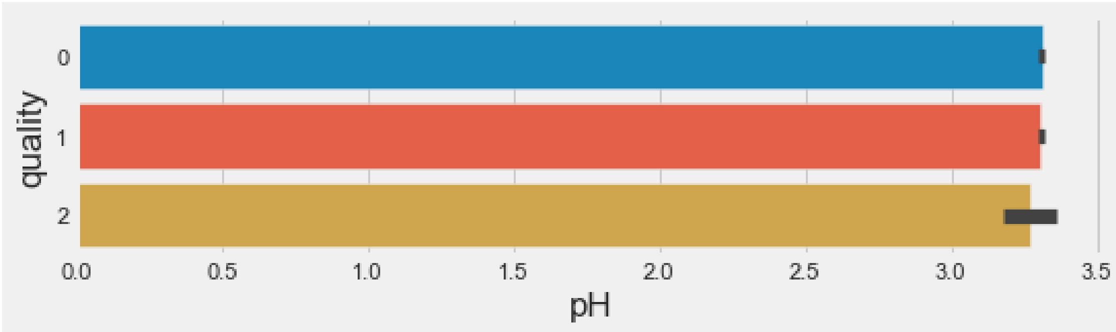

0
1
2
quality
0.0
0.5
1.0
1.5
2.0
2.5
3.0
residual sugar

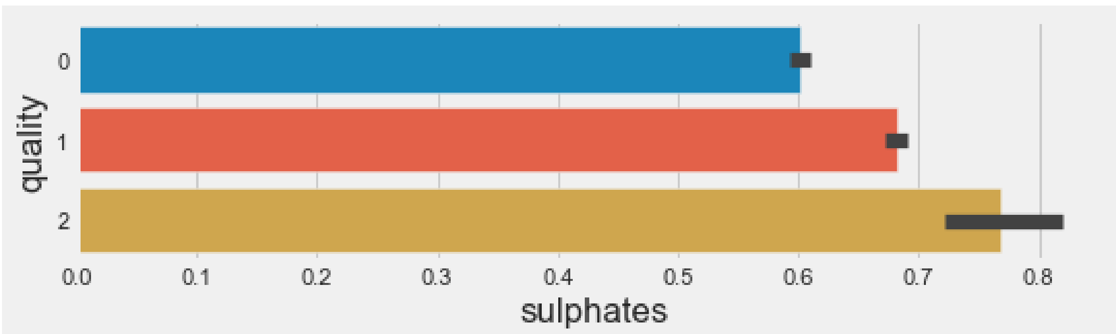

quality
0
1
2
0.00
0.02
0.04
0.06
0.08
chlorides

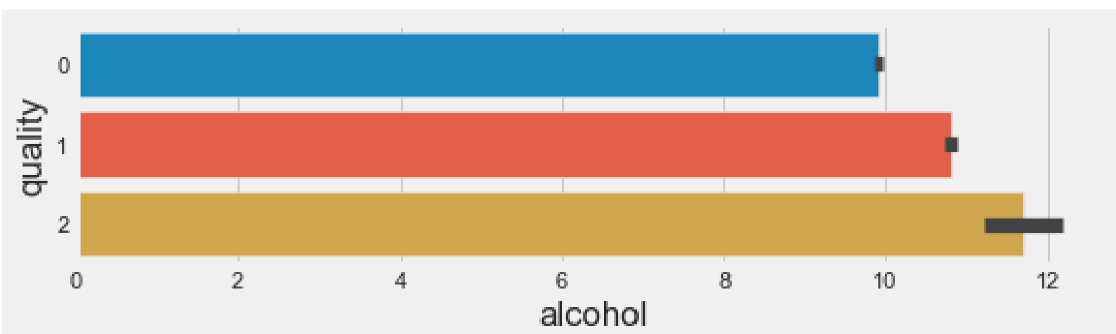

quality
0
1
2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
free sulfur dioxide


quality
0
1
2
0
10
20
30
40
50
total sulfur dioxide


quality
0
1
2
0.0
0.2
0.4
0.6
0.8
1.0
density







4 Data Preparation for model building

[40]:
```
# Defining Features X
X = attribute_only
```

[41]:
```
# Defining target y
y = wine_data[['quality']]
```

[42]:
```
# Data Standardization
X = preprocessing.StandardScaler().fit(X).transform(X)
```

[48]:
```
# Splitting data into train and test

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,␣
↪random_state=0)
```

4.0.1 Model 1 - DecisionTree Classifier

[49]:
```
# Defining Model
DT = DecisionTreeClassifier(random_state=5)
```

[50]:
```
# Fitting Model
DT.fit(X_train, y_train)
```

[50]: DecisionTreeClassifier(random_state=5)

[52]:
```
# Prediction for test data
Predicted_DT = DT.predict(X_test)
```

[53]:
```
# Checking performance of Model
accuracy_score(y_test,Predicted_DT)
```

```
              precision    recall  f1-score   support

           0       0.74      0.78      0.76       148
           1       0.78      0.73      0.75       169
           2       0.00      0.00      0.00         3

    accuracy                           0.75       320
   macro avg       0.51      0.50      0.50       320
weighted avg       0.75      0.75      0.75       320
```

[54]:
```
print(metrics.classification_report(y_test, Predicted_DT))
```


```
              precision    recall  f1-score   support

           0       0.74      0.78      0.76       148
           1       0.78      0.73      0.75       169
           2       0.00      0.00      0.00         3

    accuracy                           0.75       320
   macro avg       0.51      0.50      0.50       320
```

```
weighted avg       0.75      0.75      0.75       320
```

[58]:
```
# Prediction of Train data
Predicted_DT2 = DT.predict(X_train)
```

[59]:
```
# Checking Accuracy for Train dat a
accuracy_score(y_train,Predicted_DT2)
```

[59]:
```
1.0
```

4.0.2 Model 2 - Logistic Regression

[60]:
```
# Defining Model
LR = LogisticRegression(multi_class='multinomial',solver ='newton-cg')
```

[61]:
```
# Fitting Model
LR.fit(X_train, y_train)
```

[61]:
```
LogisticRegression(multi_class='multinomial', solver='newton-cg')
```

[62]:
```
# Prediction for test data
Predicted_LR = LR.predict(X_test)
```

[63]:
```
# Checking performance of Model
accuracy_score(y_test,Predicted_LR)
```

[63]:
```
0.740625
```

[64]:
```
print(metrics.classification_report(y_test, Predicted_LR))
```

```
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       148
           1       0.74      0.78      0.76       169
           2       0.00      0.00      0.00         3

    accuracy                           0.74       320
   macro avg       0.49      0.50      0.50       320
weighted avg       0.73      0.74      0.74       320
```

4.0.3 Model 3 - Random Forest Classifier

[67]:
```
# Defining Model
RF = RandomForestClassifier()
```

[68]:
```
# Fitting Model
RF.fit(X_train, y_train)
```

[68]: RandomForestClassifier()

[69]:
```
# Prediction for test data
Predicted_RF = RF.predict(X_test)
```

[70]:
```
# Checking performance of Model
accuracy_score(y_test,Predicted_RF)
```

[70]: 0.825

[71]:
```
print(metrics.classification_report(y_test, Predicted_RF))
```

```
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       148
           1       0.83      0.84      0.84       169
           2       0.00      0.00      0.00         3

    accuracy                           0.82       320
   macro avg       0.55      0.55      0.55       320
weighted avg       0.82      0.82      0.82       320
```