White wine

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import pearsonr

url='http://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-white.csv'
df=pd.read_csv(url,sep=';')

df.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0            7.0              0.27         0.36            20.7
0.045
1            6.3              0.30         0.34             1.6
0.049
2            8.1              0.28         0.40             6.9
0.050
3            7.2              0.23         0.32             8.5
0.058
4            7.2              0.23         0.32             8.5
0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 45.0                 170.0   1.0010  3.00       0.45

1                 14.0                 132.0   0.9940  3.30       0.49

2                 30.0                  97.0   0.9951  3.26       0.44

3                 47.0                 186.0   0.9956  3.19       0.40

4                 47.0                 186.0   0.9956  3.19       0.40


   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6
```

```
#descriptive statistics
df.describe()
```

```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4898.000000       4898.000000  4898.000000     4898.000000
mean        6.854788          0.278241     0.334192        6.391415
std         0.843868          0.100795     0.121020        5.072058
min         3.800000          0.080000     0.000000        0.600000
25%         6.300000          0.210000     0.270000        1.700000
50%         6.800000          0.260000     0.320000        5.200000
75%         7.300000          0.320000     0.390000        9.900000
max        14.200000          1.100000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide
density  \
count  4898.000000          4898.000000           4898.000000
4898.000000
mean      0.045772            35.308085            138.360657
0.994027
std       0.021848            17.007137             42.498065
0.002991
min       0.009000             2.000000              9.000000
0.987110
25%       0.036000            23.000000            108.000000
0.991723
50%       0.043000            34.000000            134.000000
0.993740
75%       0.050000            46.000000            167.000000
0.996100
max       0.346000           289.000000            440.000000
1.038980

                pH    sulphates      alcohol      quality
count  4898.000000  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267     5.877909
std       0.151001     0.114126     1.230621     0.885639
min       2.720000     0.220000     8.000000     3.000000
25%       3.090000     0.410000     9.500000     5.000000
50%       3.180000     0.470000    10.400000     6.000000
75%       3.280000     0.550000    11.400000     6.000000
max       3.820000     1.080000    14.200000     9.000000
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         4898 non-null   float64
```

```
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

```
df.isnull().sum()
```

```
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
```

Description of Qualities

1.Alcohol:the amount of alcohol in wine
2.Volatile acidity:acetic acid content which leading to an unpleasant vinegar taste
3.Sulphates:a wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant
4.Citric Acid:acts as a preservative to increase acidity for freshness and flavor to wines
5.Total Sulfur Dioxide is the amount of SO2
6.Density:sweeter wines have a higher density
7.Chlorides:the amount of salt
8.Fixed acidity:are non-volatile acids that do not evaporate easily
9.pH:the level of acidity
10.Free Sulfur Dioxide:it prevents microbial growth and the oxidation of wine 11.Residual sugar:is the amount remaining after fermentation stops

importing libraries for graphical and visualization

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

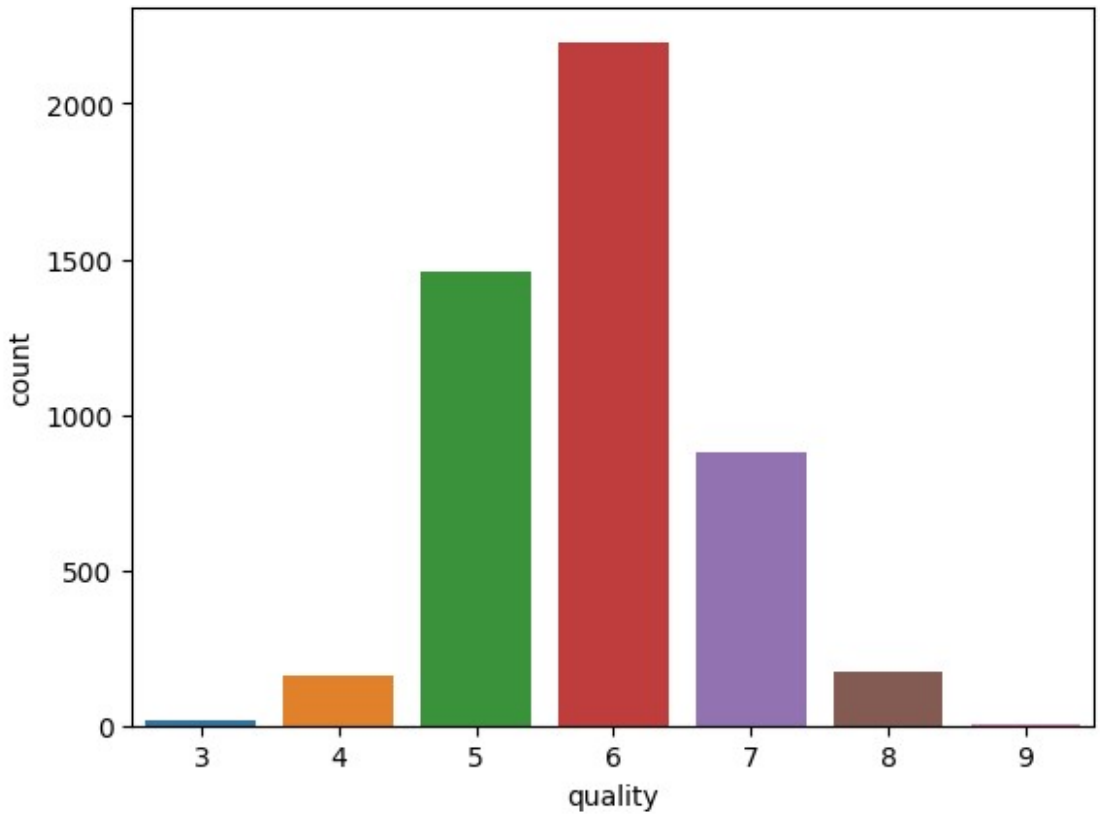
```
df.quality.unique()
```

```
array([6, 5, 7, 8, 4, 3, 9])
```

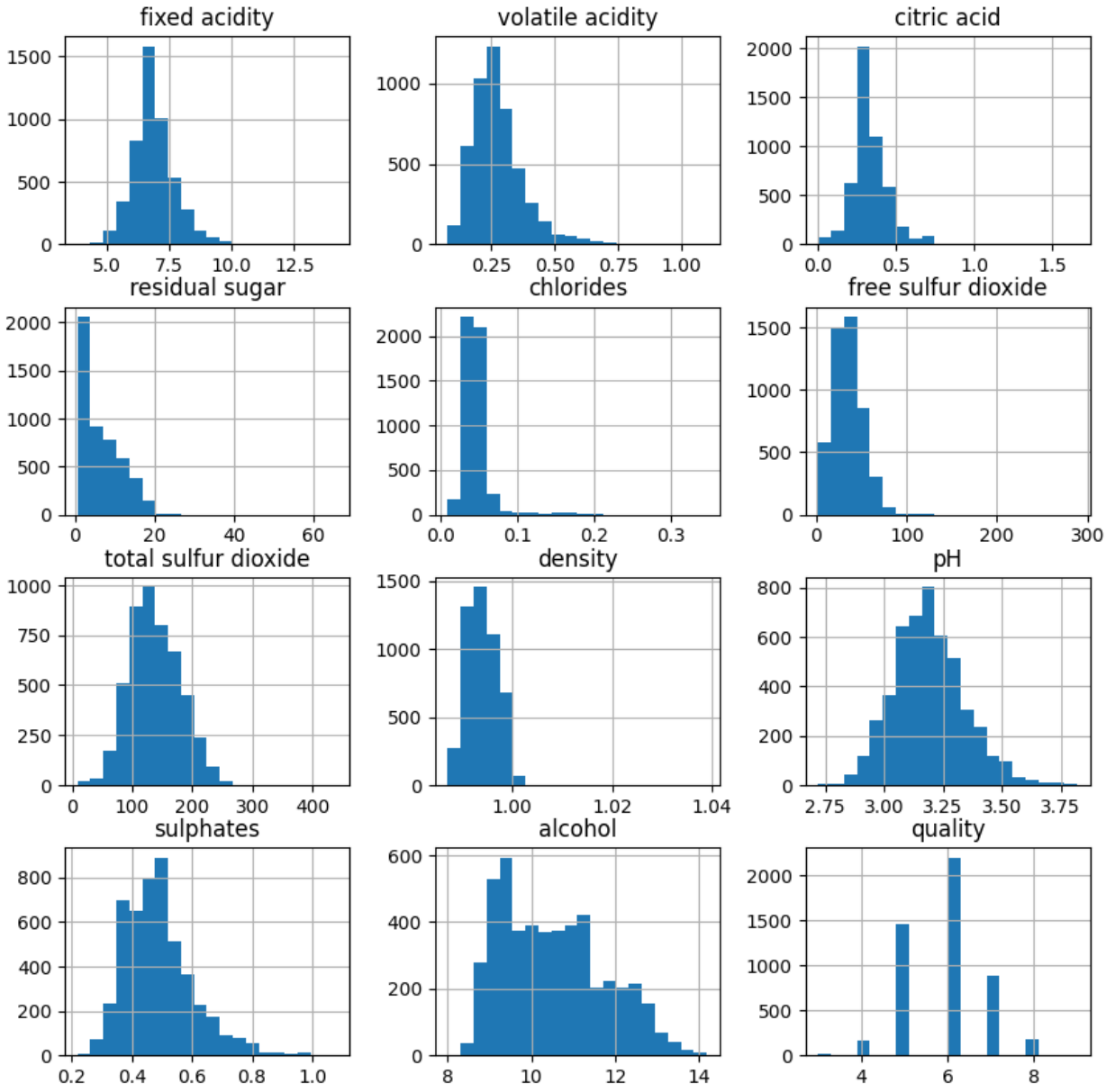
```
df.quality.value_counts()
```

```
6    2198
5    1457
7     880
8     175
4     163
3      20
9       5
Name: quality, dtype: int64
```

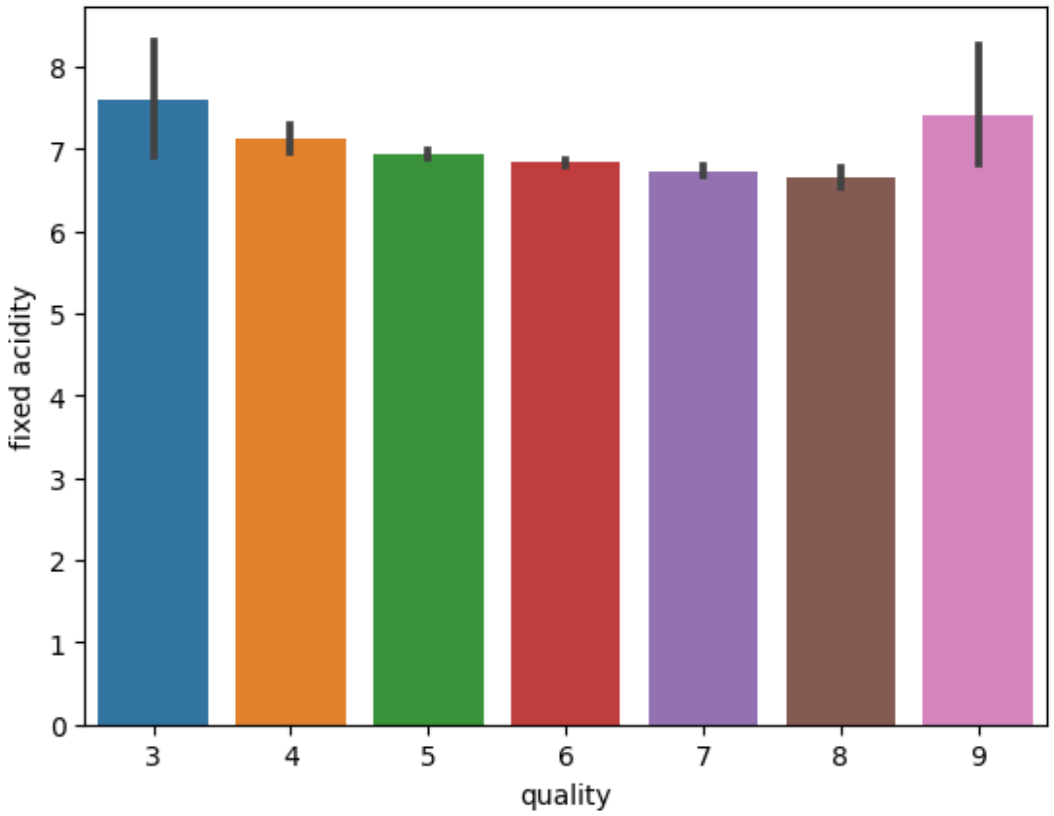
```
sns.countplot(x='quality',data=df)
plt.show()
```

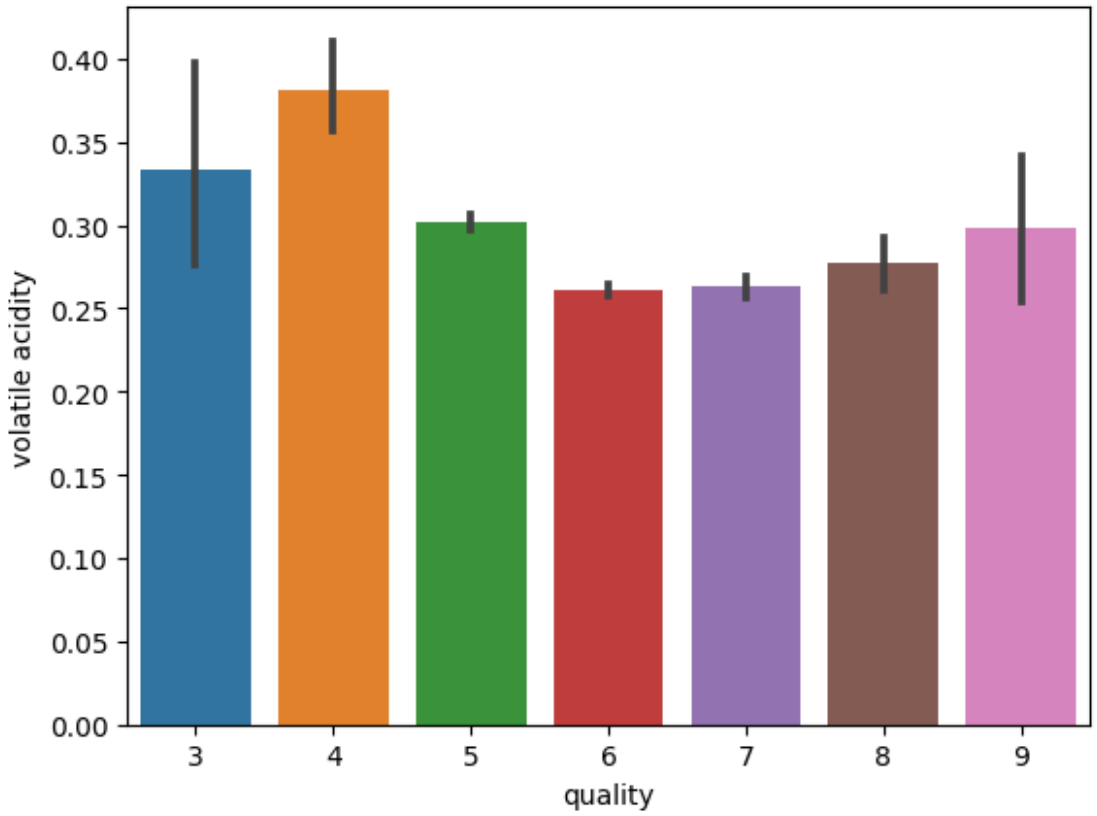


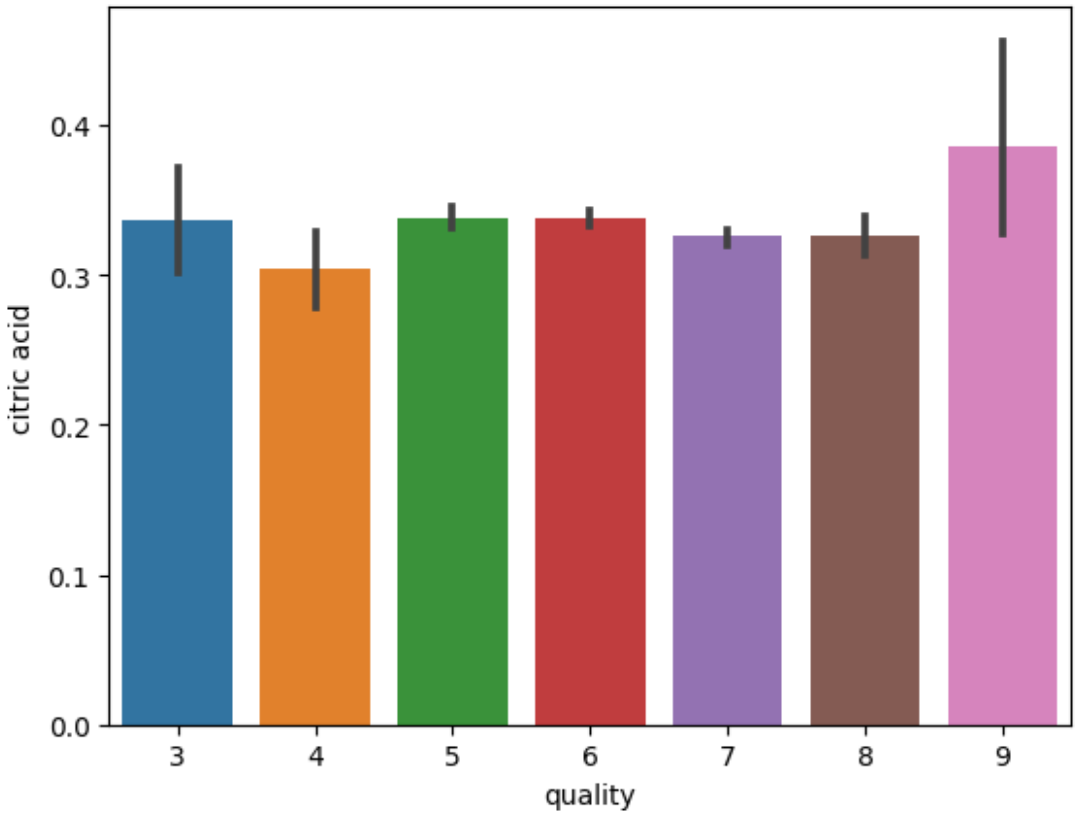

```
# histogram to visualise the distribution of the data with continuous values in the columns of the dataset.
df.hist(bins=20, figsize=(10, 10))
plt.show()
```

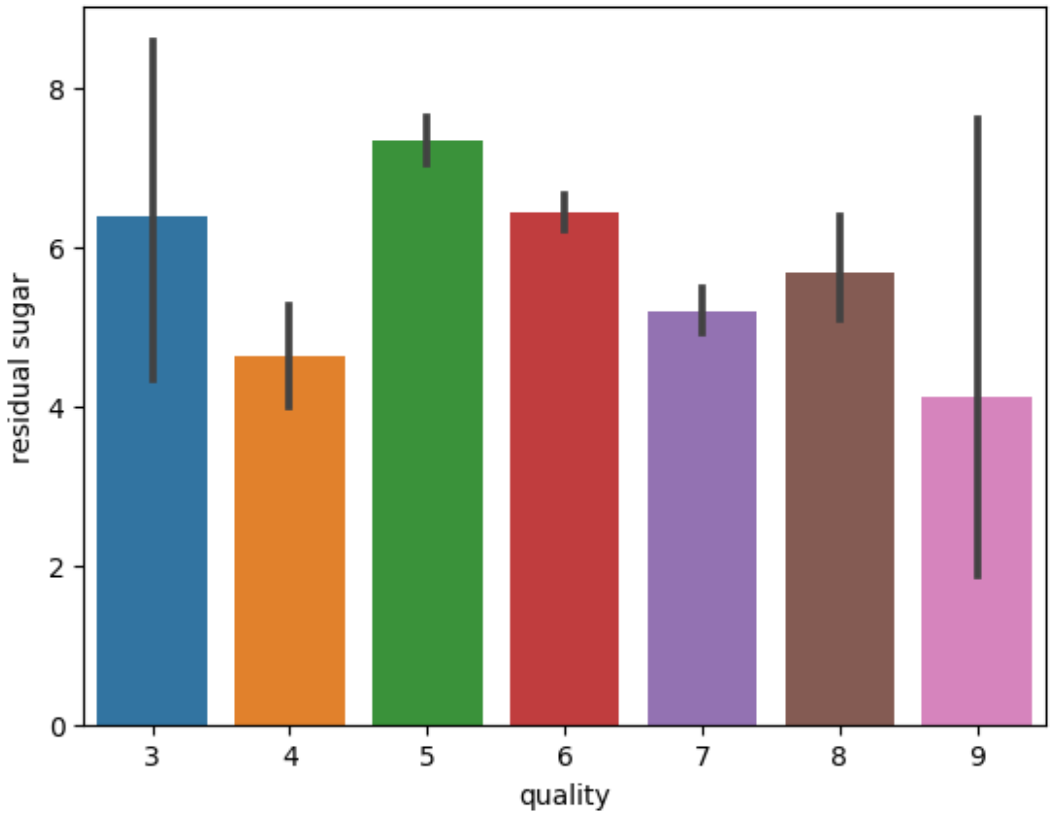


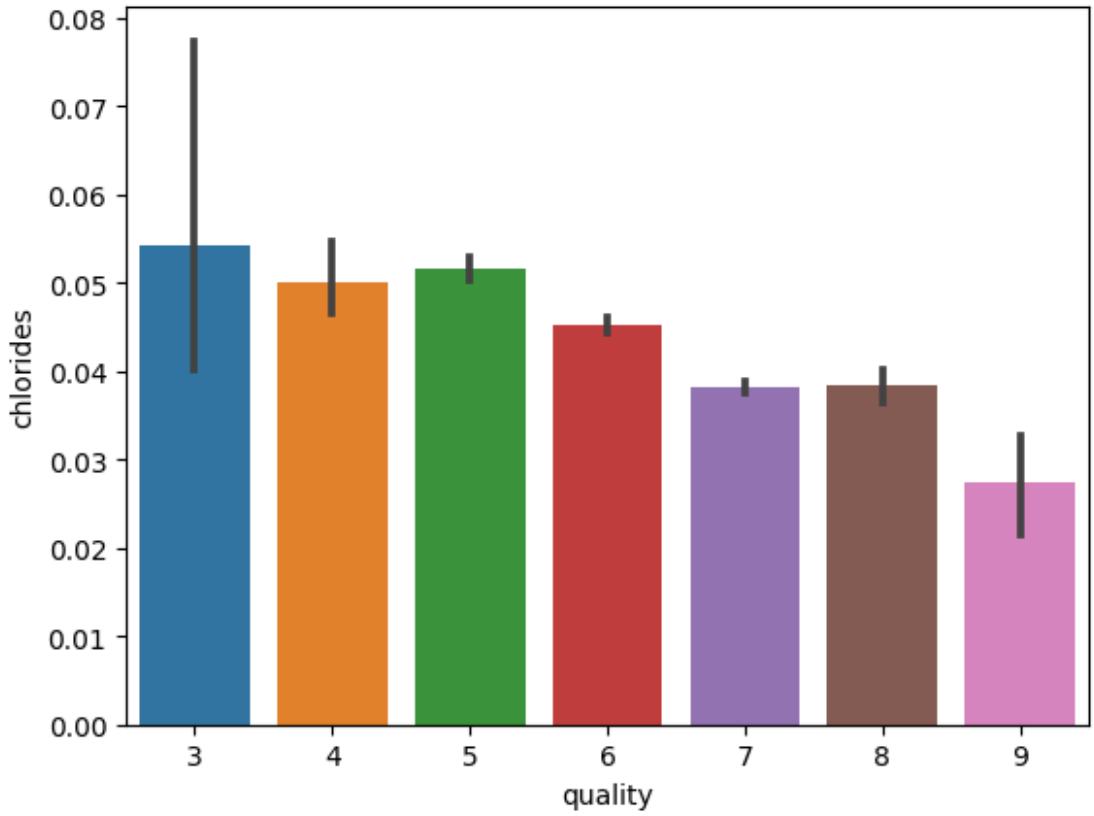

```
#df1=df.select_dtypes([np.int(), np.float()])
for i,col in enumerate(df.columns):
    plt.figure(i)
    sns.barplot(x='quality',y=col,data=df)
```

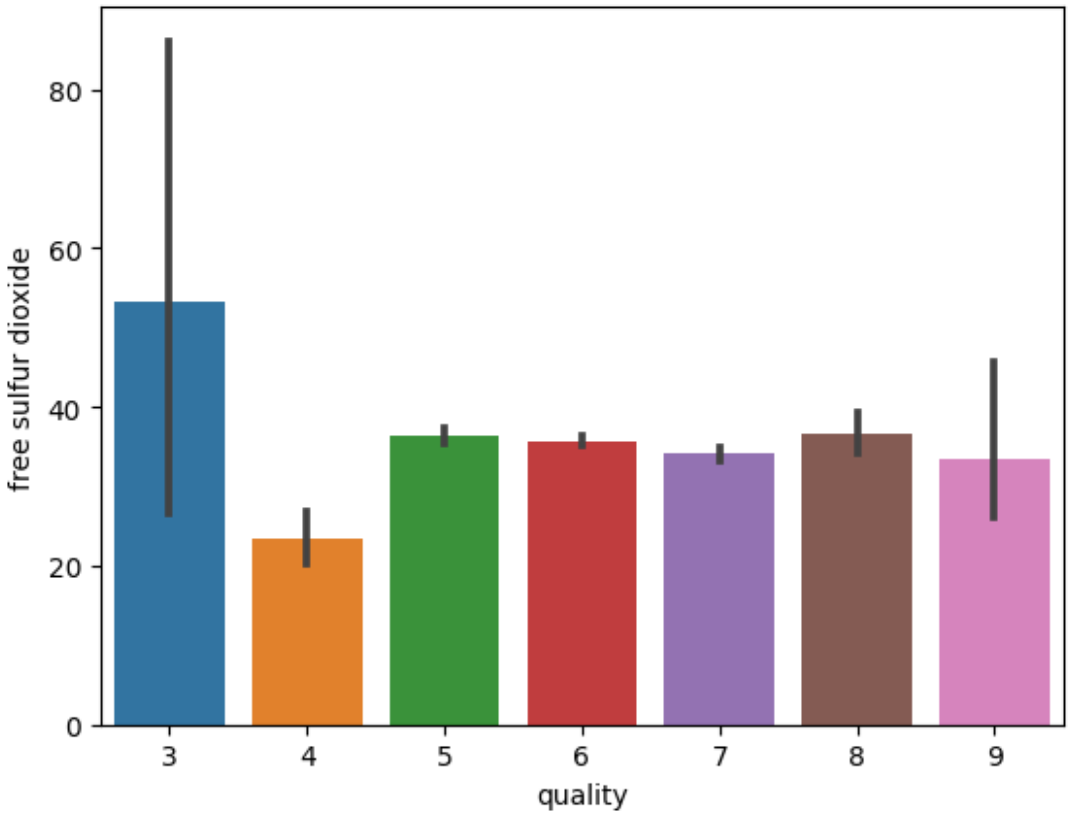

fixed acidity
0
1
2
3
4
5
6
7
8
3
4
5
6
7
8
9
quality

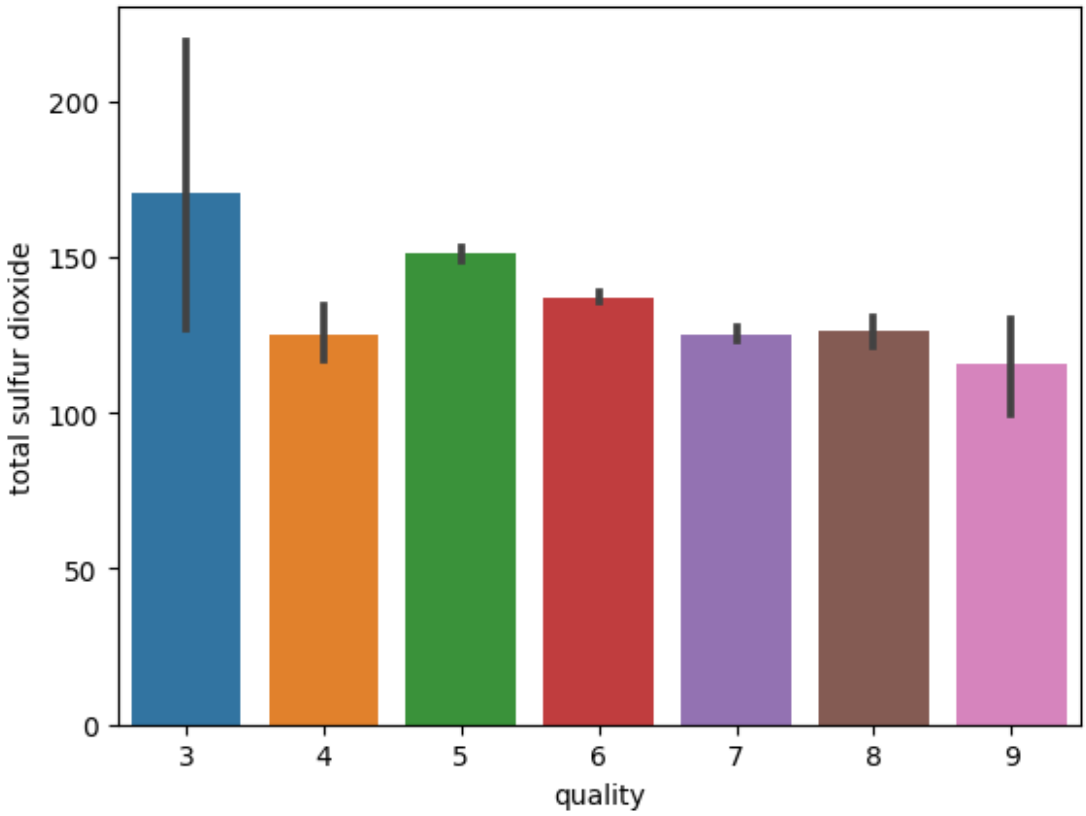

volatile acidity
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
3
4
5
6
7
8
9
quality

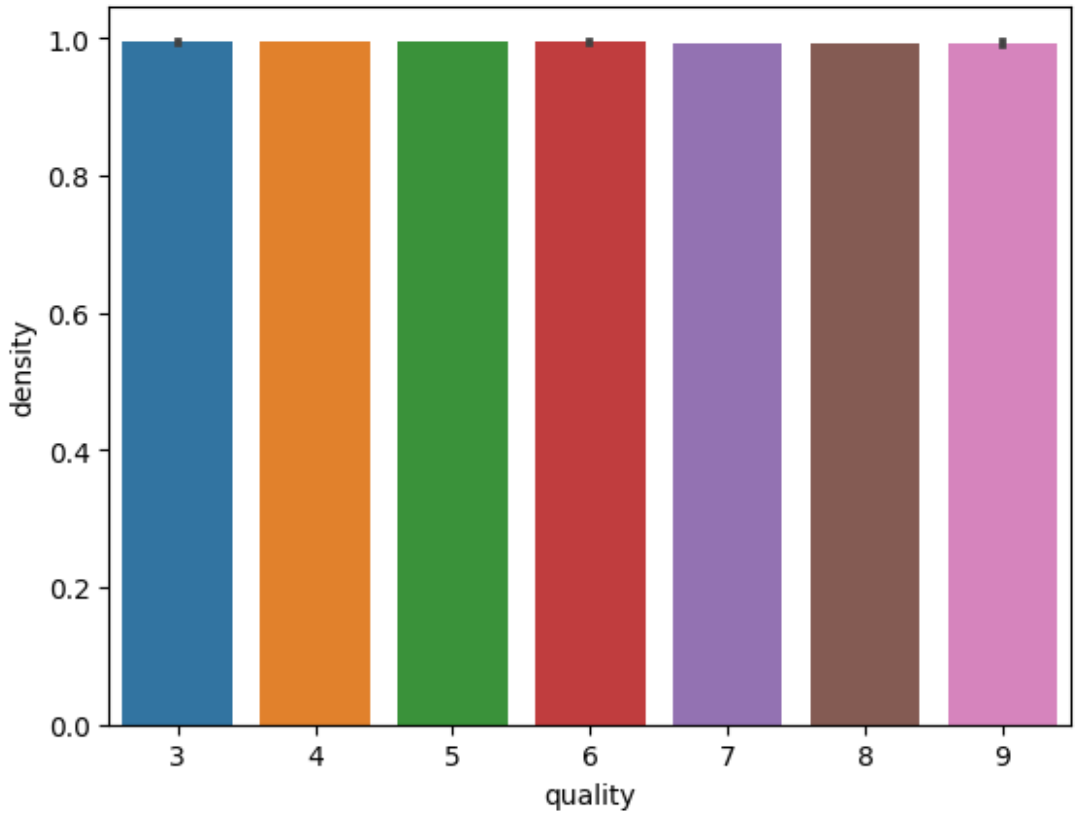

citric acid
0.0
0.1
0.2
0.3
0.4
3
4
5
6
7
8
9
quality

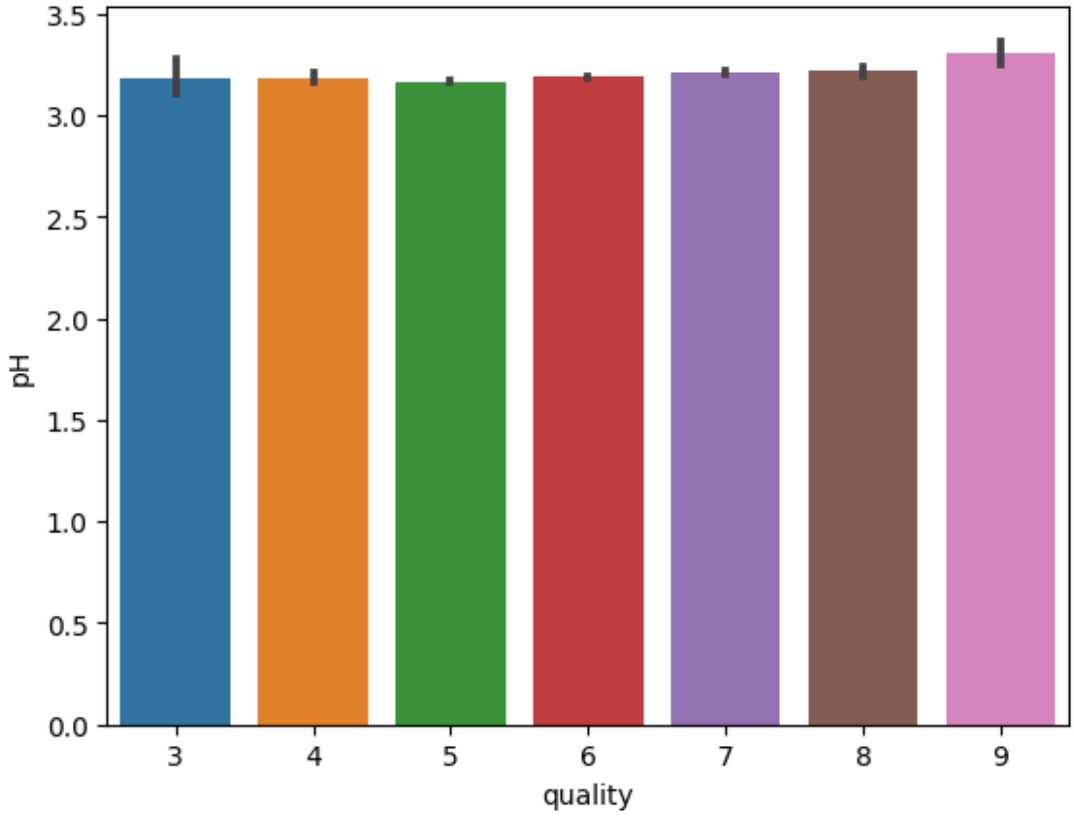

residual sugar
0
2
4
6
8
3
4
5
6
7
8
9
quality

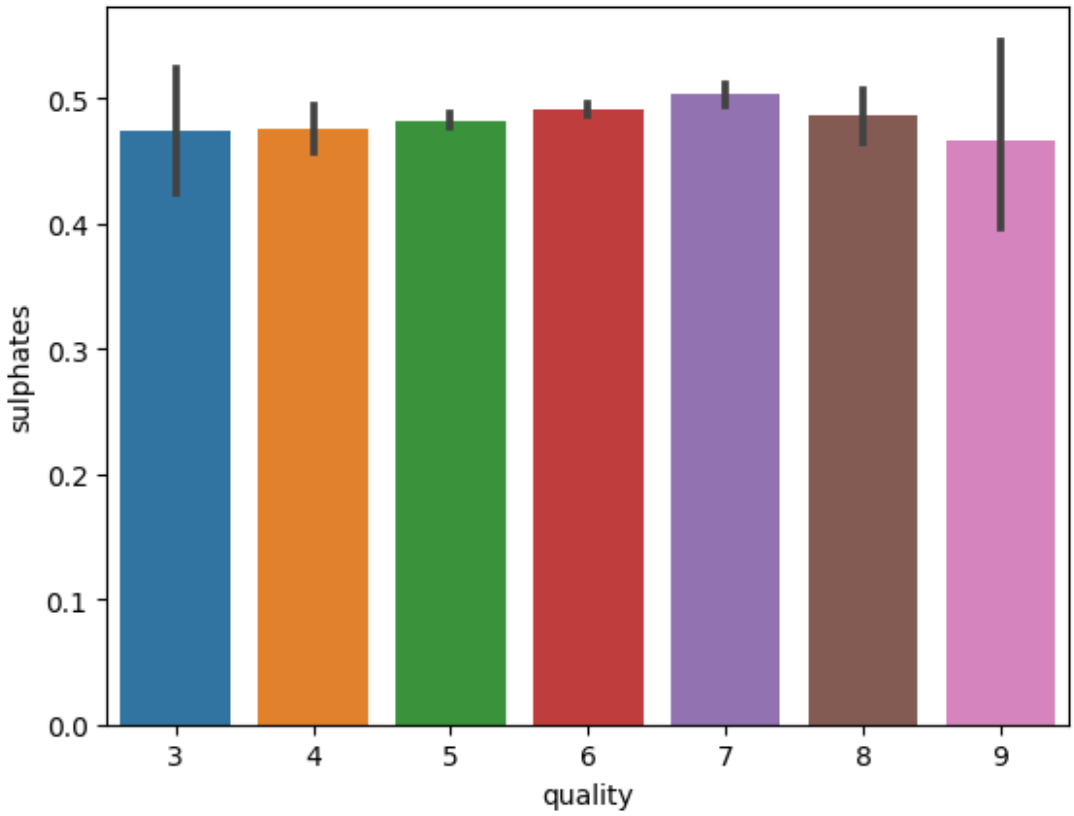

0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
chlorides
3
4
5
6
7
8
9
quality

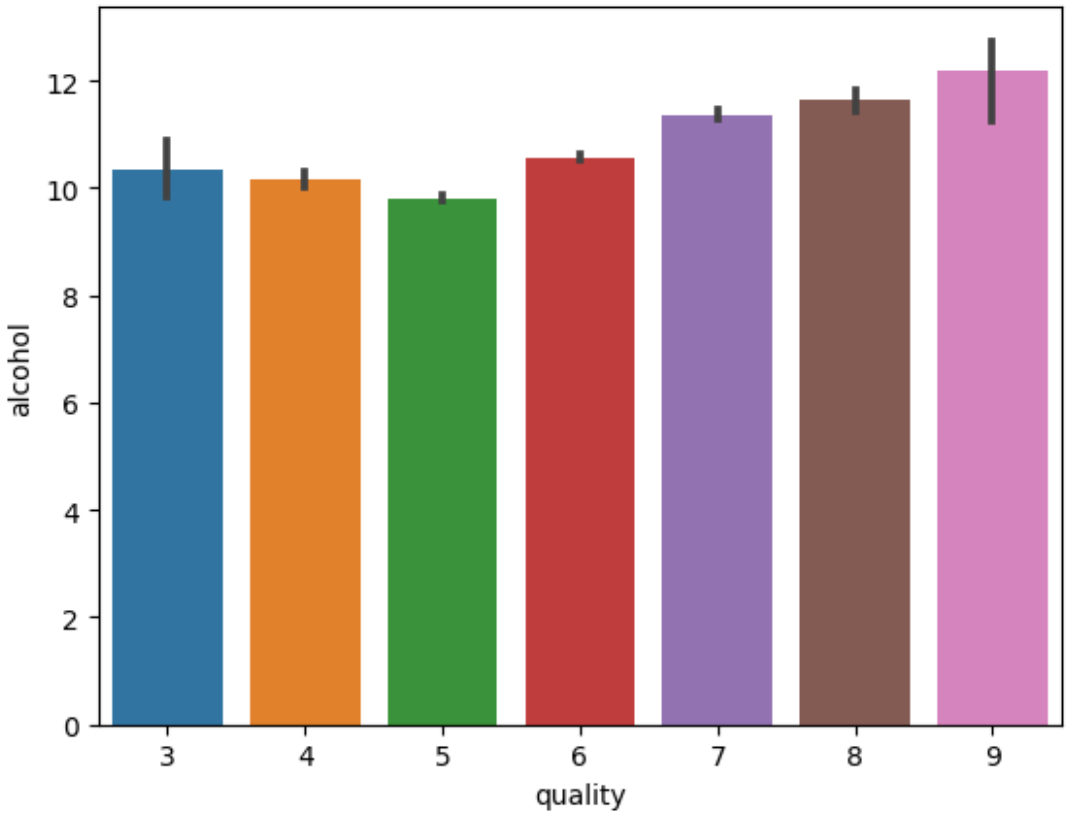

80
60
40
20
0
free sulfur dioxide
3
4
5
6
7
8
9
quality

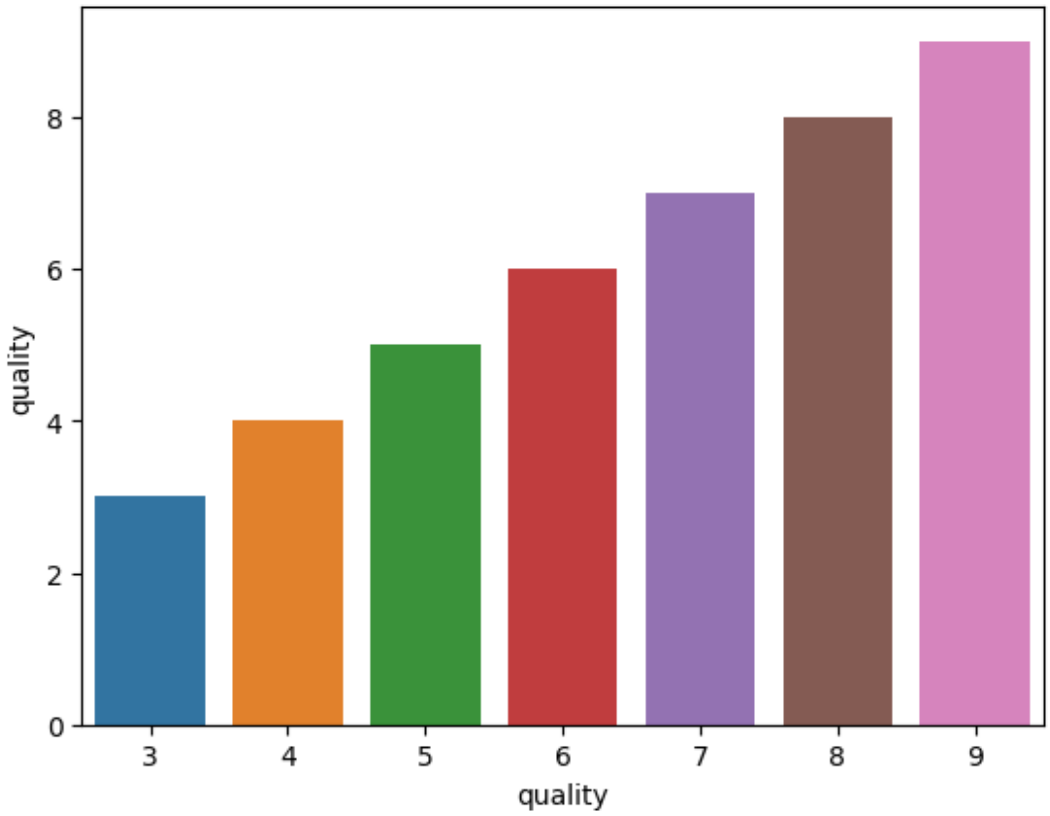

total sulfur dioxide
200
150
100
50
0
3
4
5
6
7
8
9
quality


density
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
9
quality


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
pH
3
4
5
6
7
8
9
quality

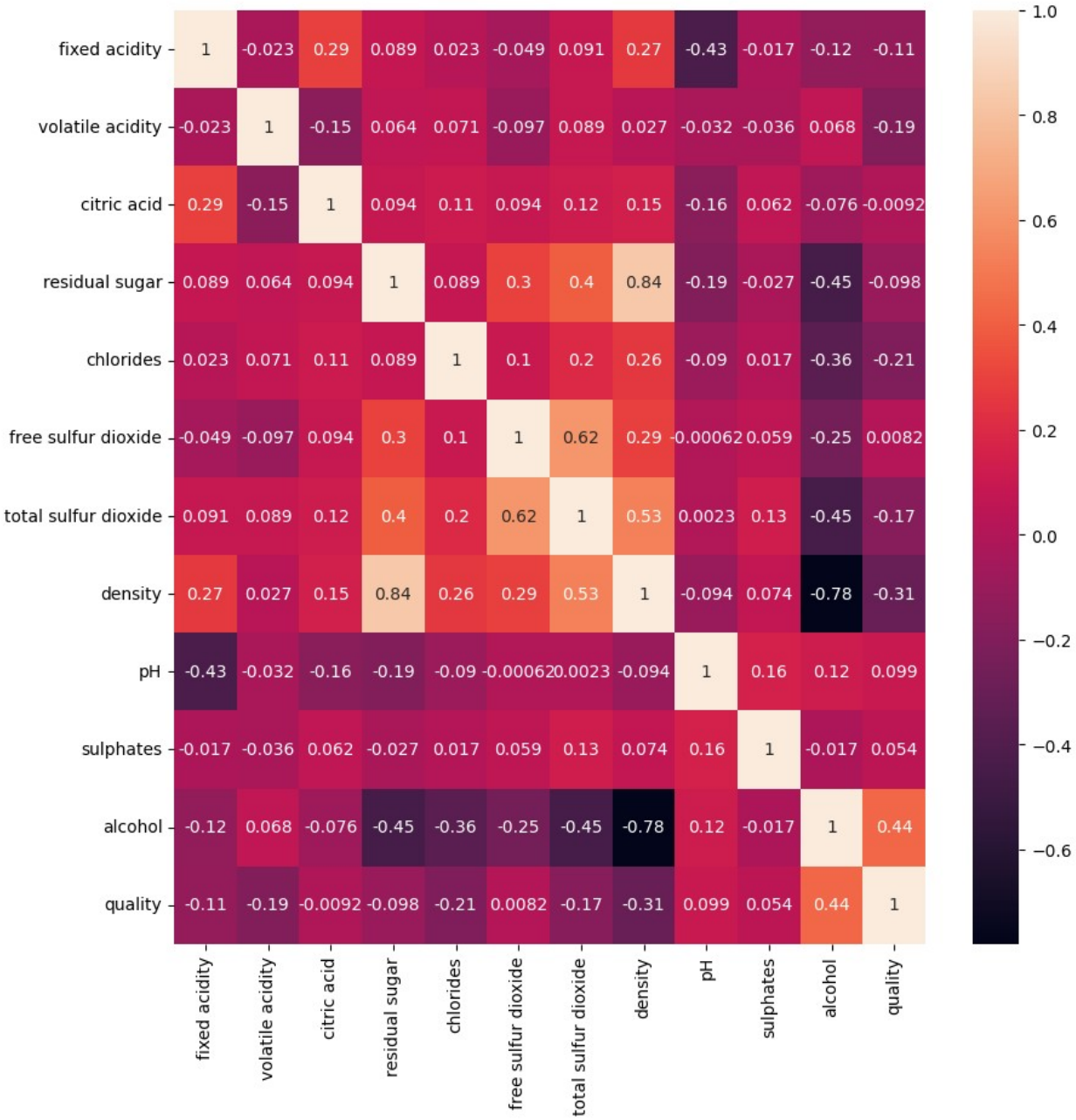

0.5
0.4
0.3
0.2
0.1
0.0
sulphates
3
4
5
6
7
8
9
quality


alcohol
quality
0
2
4
6
8
10
12
3
4
5
6
7
8
9


quality
quality
0
2
4
6
8
3
4
5
6
7
8
9

```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),color='k',annot=True)
```

```
<Axes: >
```

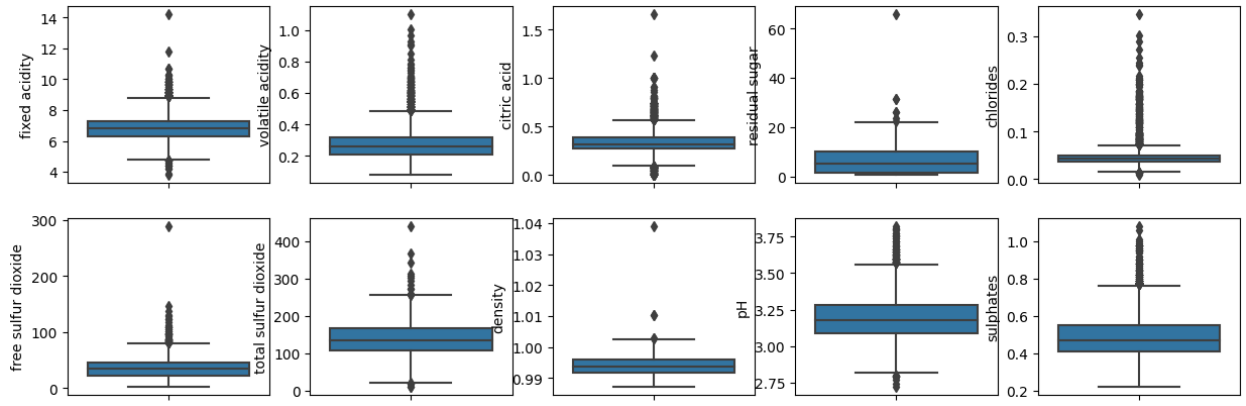


Checking for outliers in our dataset

```
fig, ax = plt.subplots(ncols=5, nrows=2, figsize=(15, 5))
ax = ax.flatten()
index = 0
for i in df.columns:
```

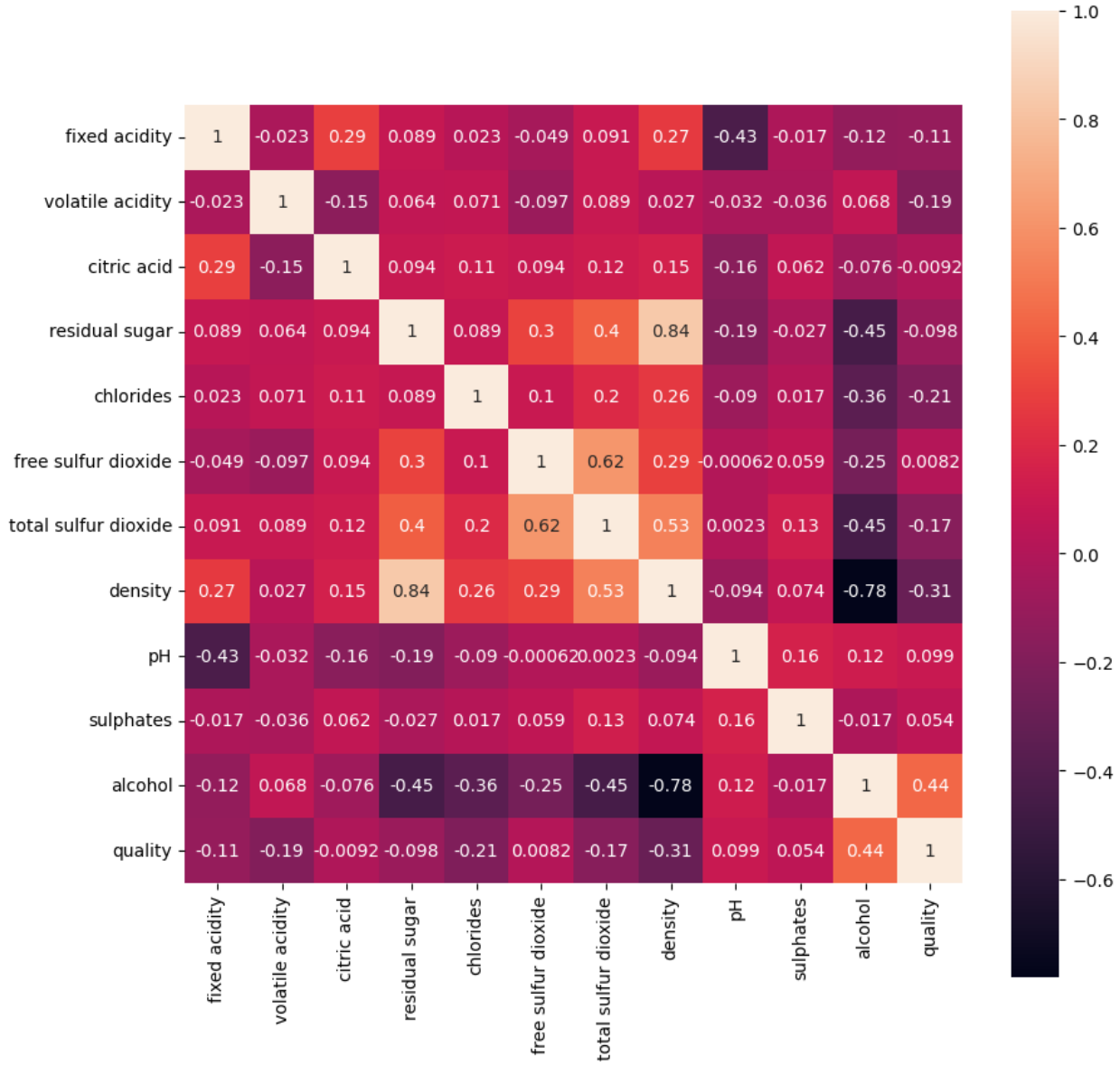
```
  if i != 'quality':
    sns.boxplot(y=i, data=df, ax=ax[index])
    index +=1
plt.tight_layout(pad=0.4)
plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-66-55e2a001decc> in <cell line: 4>()
      4 for i in df.columns:
      5   if i != 'quality':
----> 6     sns.boxplot(y=i, data=df, ax=ax[index])
      7     index +=1
      8 plt.tight_layout(pad=0.4)

IndexError: index 10 is out of bounds for axis 0 with size 10
```



From the above box plots we can clearly see that there are outliers in all features but Here I am choosing not remove/modify outliers as we are looking for accuracy to minute levels, not just some approximation — high quality wine may have very rare composition (hence outlier) from other average quality wines, so we can not remove or modify outlier values in out dataset.

```
plt.figure(figsize=(10, 10))
sns.heatmap(df.corr(method='pearson'), annot=True, square=True)
plt.show()

print('Correlation of different features of our dataset with quality:')
for i in df.columns:
  corr, _ = pearsonr(df[i], df['quality'])
  print('%s : %.4f' %(i,corr))
```




```
Correlation of different features of our dataset with quality:
fixed acidity : -0.1137
volatile acidity : -0.1947
citric acid : -0.0092
residual sugar : -0.0976
chlorides : -0.2099
free sulfur dioxide : 0.0082
total sulfur dioxide : -0.1747
density : -0.3071
pH : 0.0994
sulphates : 0.0537
alcohol : 0.4356
quality : 1.0000
```

From the above plots and values we can conclude:

volatile acidity, chlorides are negatively correlated to quality free sulfur dioxide and total sulfur dioxide are highly correlated to each other with correlation of 0.62. There are many features with correlation < 0.5 to quality, and may be removed from the dataset. BUT for the same reason as mentioned above in outlier section, that -- we are looking for accuracy to minute levels, not just some approximation — high quality wine may have very rare composition from other average quality wines, hence we need to take every feature in account while predicting quality of wine, so we can not remove or modify outlier values in out dataset.

Creating Classification bins

```
bins=(2,6.5,8)
group_names=['bad','good']
df['quality']=pd.cut(df['quality'],bins=bins,labels=group_names)

#importing sklearn packages for machine learning
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

Classification into ones and zeros using LabelEncoder()

```
# Assigning a label to our quality variable
label_quality = LabelEncoder()
df['quality'] = label_quality.fit_transform(df['quality'])

df.head(15)
```

```
    fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0             7.0              0.27         0.36           20.70
0.045
1             6.3              0.30         0.34            1.60
0.049
2             8.1              0.28         0.40            6.90
0.050
3             7.2              0.23         0.32            8.50
0.058
4             7.2              0.23         0.32            8.50
0.058
```

```
5              8.1              0.28         0.40             6.90
0.050
6              6.2              0.32         0.16             7.00
0.045
7              7.0              0.27         0.36            20.70
0.045
8              6.3              0.30         0.34             1.60
0.049
9              8.1              0.22         0.43             1.50
0.044
10             8.1              0.27         0.41             1.45
0.033
11             8.6              0.23         0.40             4.20
0.035
12             7.9              0.18         0.37             1.20
0.040
13             6.6              0.16         0.40             1.50
0.044
14             8.3              0.42         0.62            19.25
0.040

    free sulfur dioxide  total sulfur dioxide  density    pH
sulphates  \
0                  45.0                 170.0   1.0010  3.00
0.45
1                  14.0                 132.0   0.9940  3.30
0.49
2                  30.0                  97.0   0.9951  3.26
0.44
3                  47.0                 186.0   0.9956  3.19
0.40
4                  47.0                 186.0   0.9956  3.19
0.40
5                  30.0                  97.0   0.9951  3.26
0.44
6                  30.0                 136.0   0.9949  3.18
0.47
7                  45.0                 170.0   1.0010  3.00
0.45
8                  14.0                 132.0   0.9940  3.30
0.49
9                  28.0                 129.0   0.9938  3.22
0.45
10                 11.0                  63.0   0.9908  2.99
0.56
11                 17.0                 109.0   0.9947  3.14
0.53
12                 16.0                  75.0   0.9920  3.18
0.63
```

```
13                   48.0                  143.0   0.9912  3.54
0.52
14                   41.0                  172.0   1.0002  2.98
0.67

    alcohol  quality
0       8.8        0
1       9.5        0
2      10.1        0
3       9.9        0
4       9.9        0
5      10.1        0
6       9.6        0
7       8.8        0
8       9.5        0
9      11.0        0
10     12.0        0
11      9.7        0
12     10.8        0
13     12.4        1
14      9.7        0
```

Setting the dependent and independent Variables

```
Y = df.quality
X = df.drop('quality', axis=1)
```

spliting the data into training and testing data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2, random_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Building Machine Learning Models to compare

```
def models(X_train,Y_train):

  #Using Logistic Regression Algorithm to the Training Set
  from sklearn.linear_model import LogisticRegression
  log = LogisticRegression(random_state = 0)
  log.fit(X_train, Y_train)

  #Using KNeighborsClassifier Method of neighbors class to use Nearest
Neighbor algorithm
  from sklearn.neighbors import KNeighborsClassifier
  knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p
```

```
= 2)
  knn.fit(X_train, Y_train)

  #Using SVC method of svm class to use Support Vector Machine
Algorithm
  from sklearn.svm import SVC
  svc_lin = SVC(kernel = 'linear', random_state = 0)
  svc_lin.fit(X_train, Y_train)

  #Using SVC method of svm class to use Kernel SVM Algorithm
  from sklearn.svm import SVC
  svc_rbf = SVC(kernel = 'rbf', random_state = 0)
  svc_rbf.fit(X_train, Y_train)

  #Using GaussianNB method of naïve_bayes class to use Naïve Bayes
Algorithm
  from sklearn.naive_bayes import GaussianNB
  gauss = GaussianNB()
  gauss.fit(X_train, Y_train)

  #Using DecisionTreeClassifier of tree class to use Decision Tree
Algorithm
  from sklearn.tree import DecisionTreeClassifier
  tree = DecisionTreeClassifier(criterion = 'entropy', random_state =
0)
  tree.fit(X_train, Y_train)

  #Using RandomForestClassifier method of ensemble class to use Random
Forest Classification algorithm
  from sklearn.ensemble import RandomForestClassifier
  forest = RandomForestClassifier(n_estimators = 10, criterion =
'entropy', random_state = 0)
  forest.fit(X_train, Y_train)
  #print model accuracy on the training data.
  print('[0]Logistic Regression Training Accuracy:',
log.score(X_train, Y_train))
  print('[1]K Nearest Neighbor Training Accuracy:', knn.score(X_train,
Y_train))
  print('[2]Support Vector Machine (Linear Classifier) Training
Accuracy:', svc_lin.score(X_train, Y_train))
  print('[3]Support Vector Machine (RBF Classifier) Training
Accuracy:', svc_rbf.score(X_train, Y_train))
  print('[4]Gaussian Naive Bayes Training Accuracy:',
gauss.score(X_train, Y_train))
  print('[5]Decision Tree Classifier Training Accuracy:',
tree.score(X_train, Y_train))
  print('[6]Random Forest Classifier Training Accuracy:',
forest.score(X_train, Y_train))
  return log, knn, svc_lin, svc_rbf, gauss, tree, forest
```

Evaluating Performance on Training Sets

```
model = models(X_train,Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

[0]Logistic Regression Training Accuracy: 0.8006636038795304
[1]K Nearest Neighbor Training Accuracy: 0.8917815211842777
[2]Support Vector Machine (Linear Classifier) Training Accuracy:
0.7845839714139867
[3]Support Vector Machine (RBF Classifier) Training Accuracy:
0.8420112302194998
[4]Gaussian Naive Bayes Training Accuracy: 0.7240939254721797
[5]Decision Tree Classifier Training Accuracy: 1.0
[6]Random Forest Classifier Training Accuracy: 0.9892802450229708
```

Random Forrest Classification model gave the best accuracy and can be considered as a good model for predictiong the quality of wine for this problem. However other models like Logisgic Regression, KNN and SVC also have comparable score to Random Forrest and may also be used to predict quality of wine. Naive Bayes model gave the least accuracy, which can be considered bad model to predict the quality of fine.