imports

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import roc_curve,confusion_matrix, classification_report , accuracy_sc
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV, cross_val_score
```

In [2]:

```
data=pd.read_csv('QualityPrediction.csv')
data
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	

1599 rows × 12 columns

EDA

In [3]:

```
data.describe()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total su dio
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000

In [4]:

```
data.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 [5]:

```
data['quality'].value_counts
```

Out[5]:

```
<bound method IndexOpsMixin.value_counts of 0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64>
```

Data Visualization

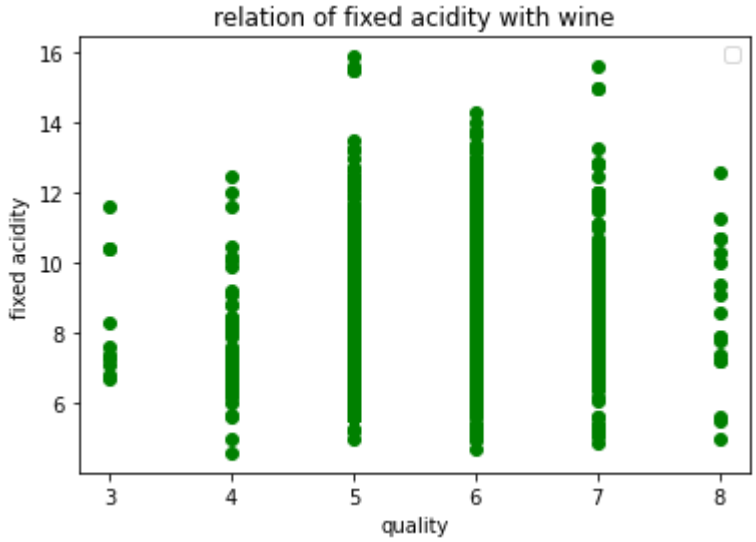
Bivariate Analysis

In [6]:

```
# checking the variation of fixed acidity in the different qualities of wine

plt.scatter(data['quality'], data['fixed acidity'], color = 'green')
plt.title('relation of fixed acidity with wine')
plt.xlabel('quality')
plt.ylabel('fixed acidity')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```

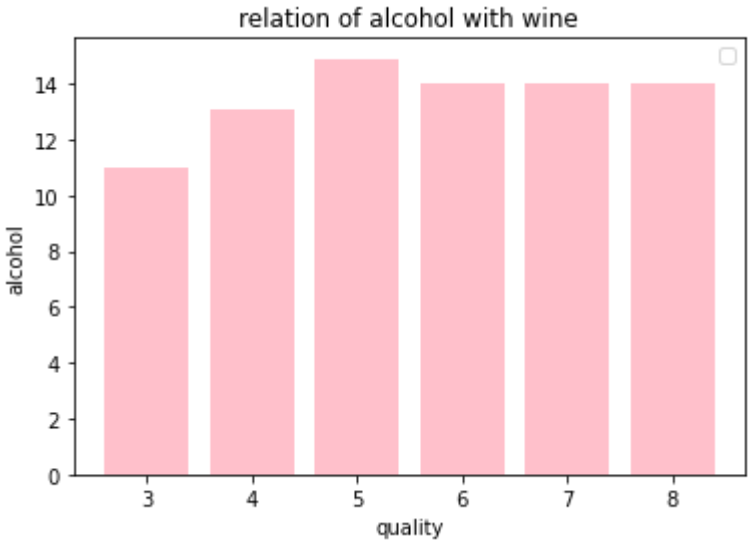


In [7]:

```
# checking the variation of fixed acidity in the different qualities of wine

plt.bar(data['quality'], data['alcohol'], color = 'pink')
plt.title('relation of alcohol with wine')
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



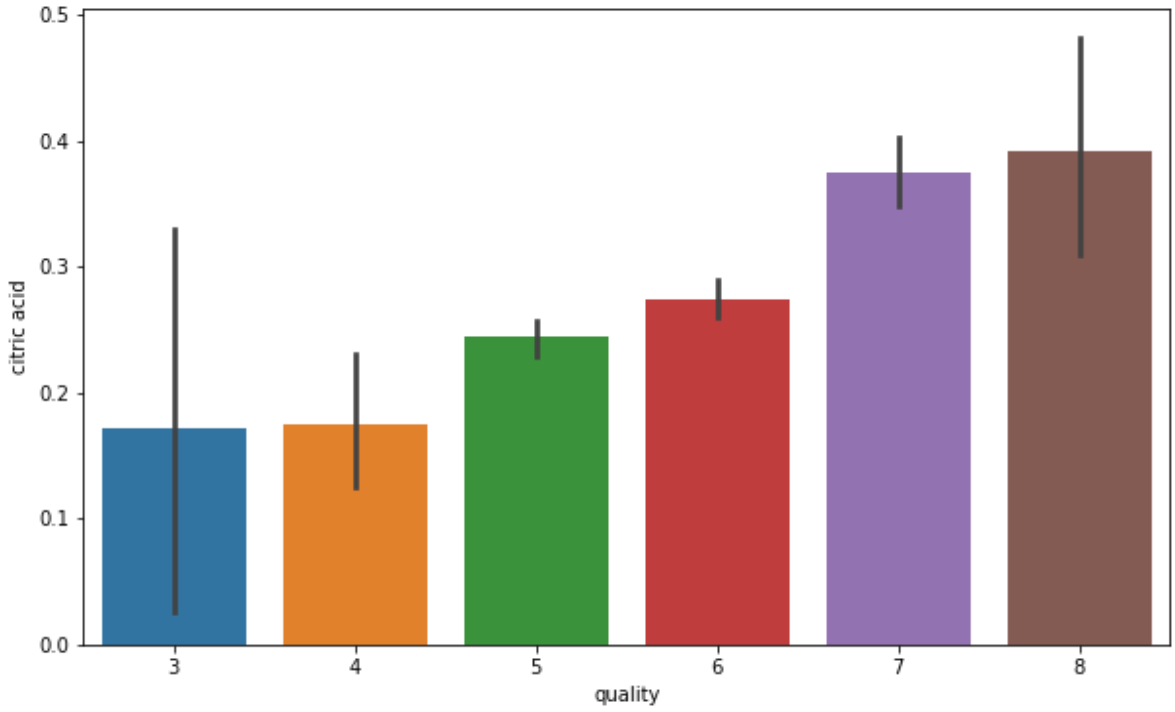
In [8]:

```
import seaborn as sns

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = data)
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2599dc78250>
```

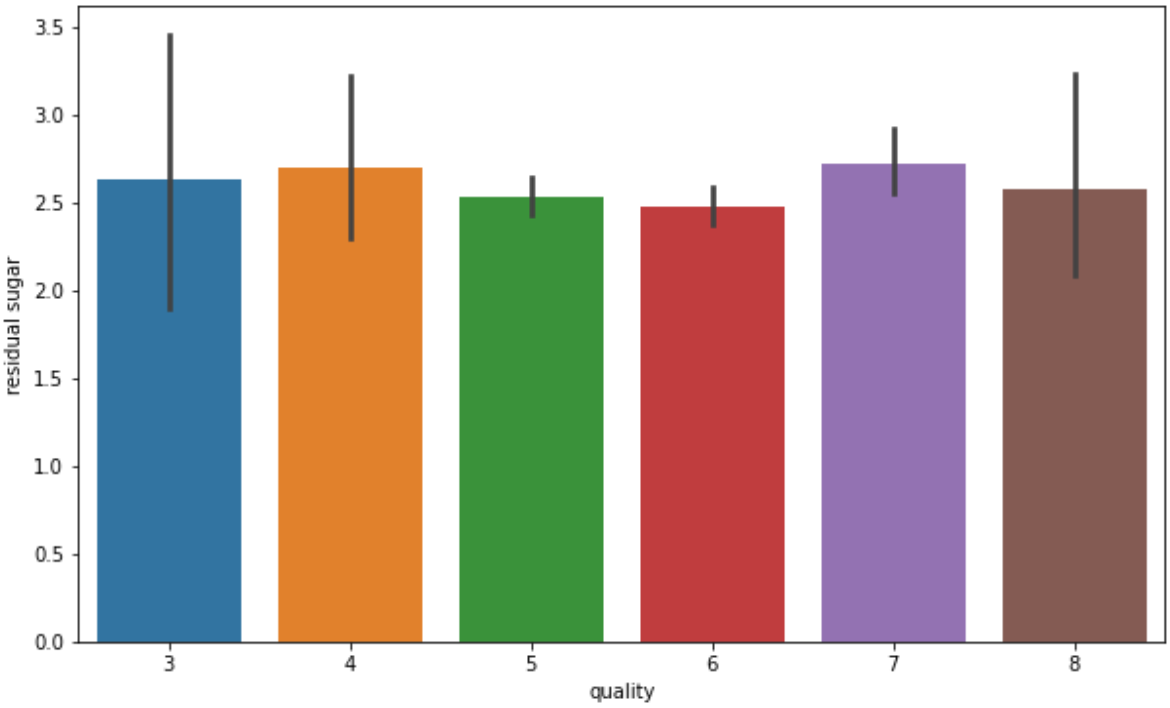


In [9]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = data)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2599ccda490>
```

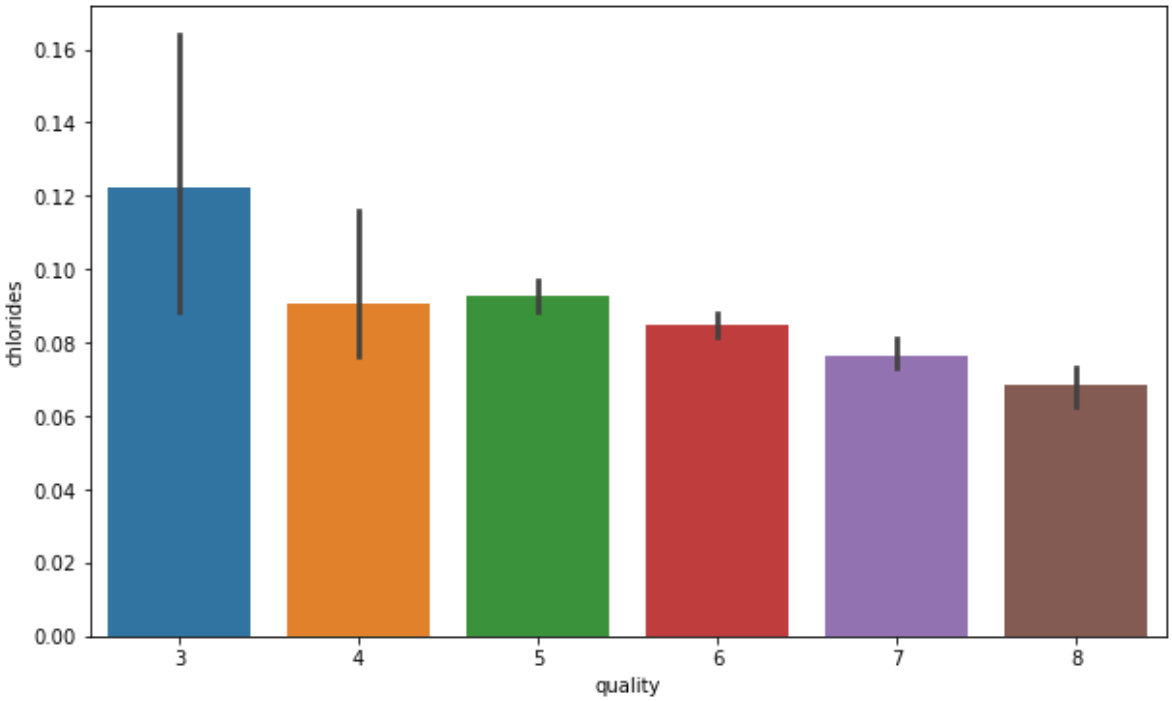


In [10]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = data)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2599d7d5940>
```

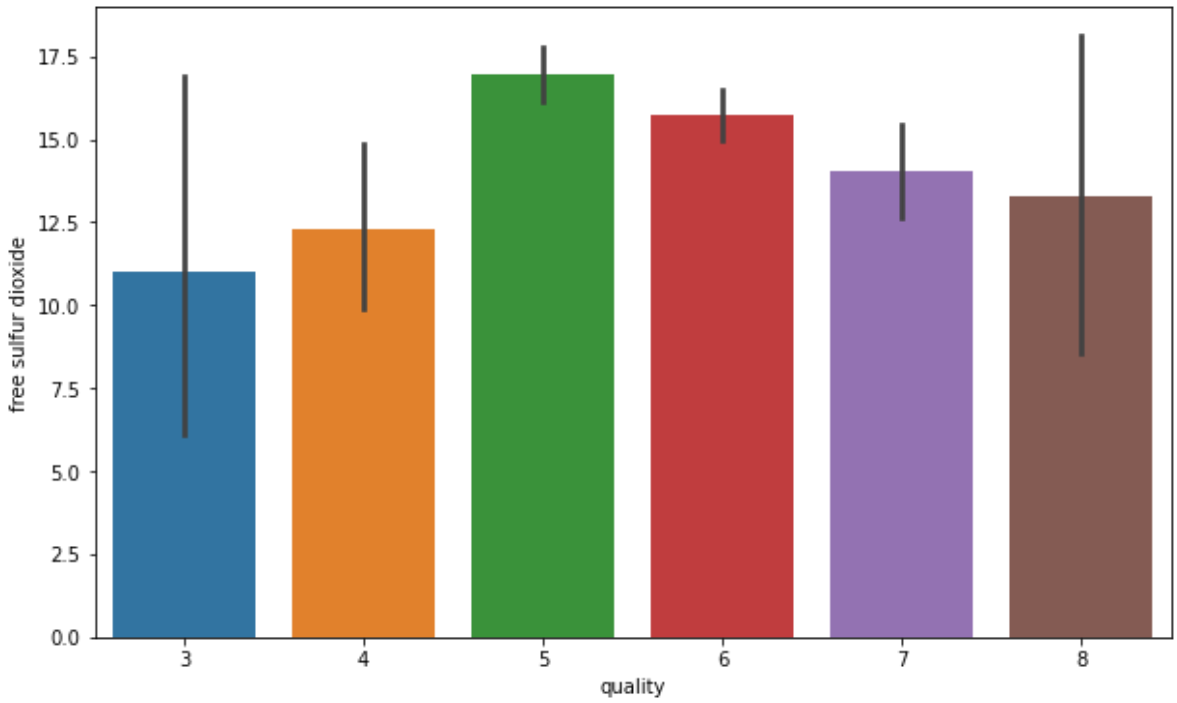


In [11]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = data)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2599ddc25b0>
```



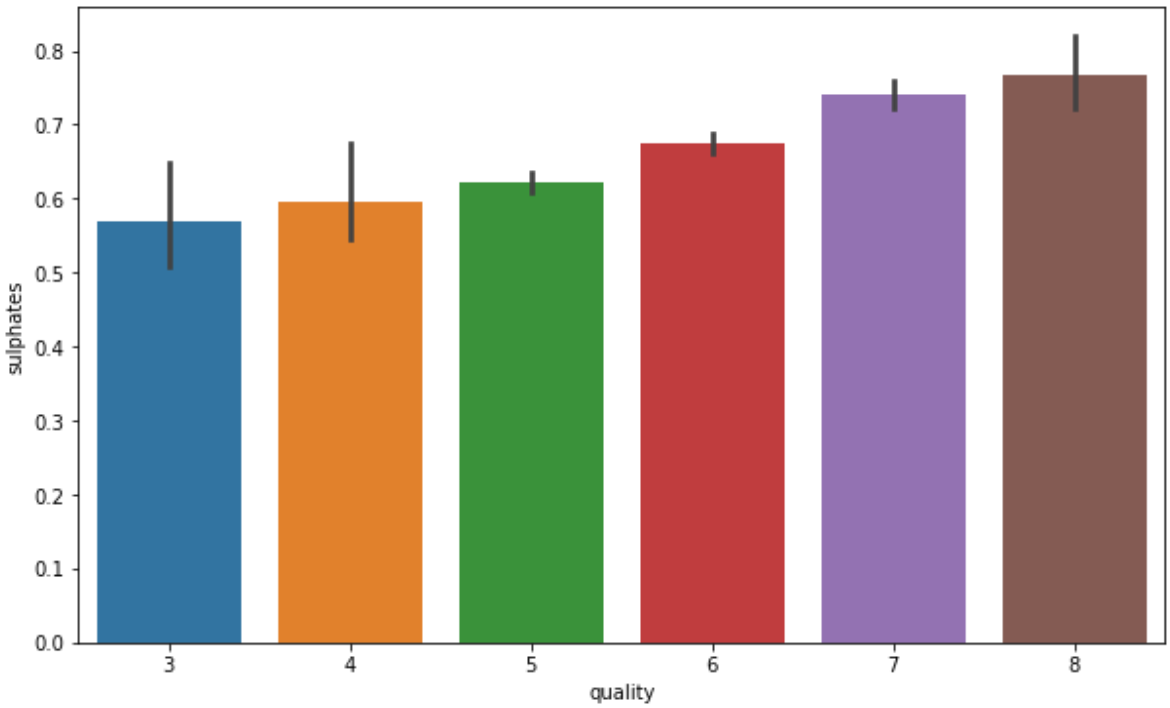
In [12]:

```
#Sulphates level goes higher with the quality of wine

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = data)
```

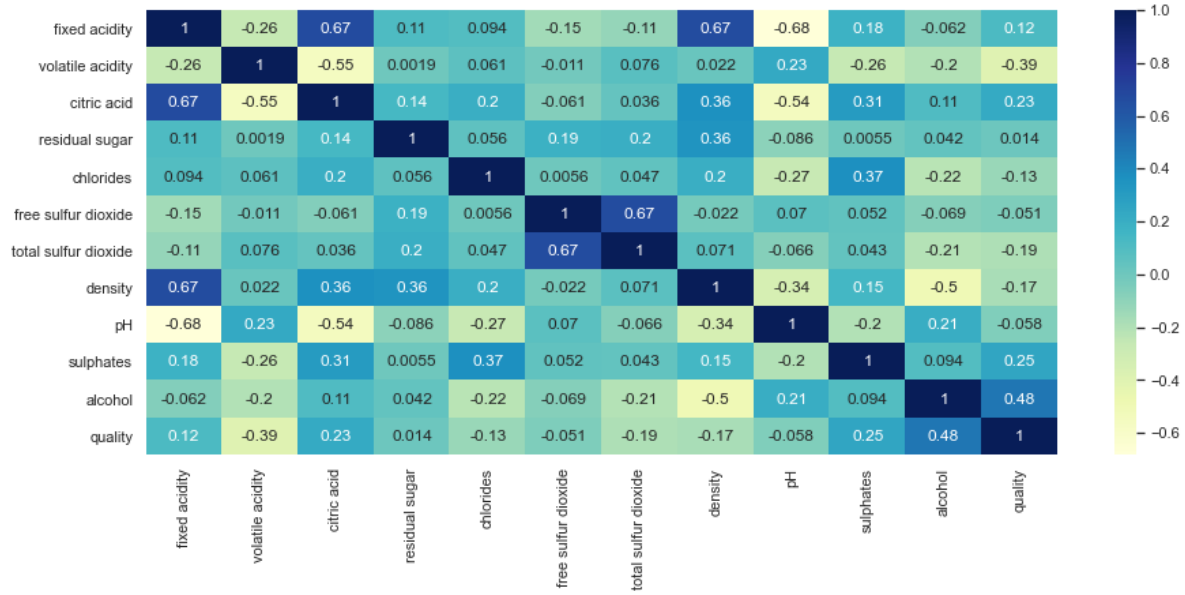
Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2599ddd0040>
```



In [13]:

```
sns.set(rc={'figure.figsize':(15,6)})
sns.heatmap(data.corr(), cmap = "YlGnBu", annot =True)
plt.show()
```



From the above correlation plot for the given dataset for wine quality prediction, we can easily see which items are related strongly with each other and which items are related weekly with each other. For Example,

The strongly correlated items are : 1.fixed acidity and citric acid. 2.free sulphur dioxide and total sulphor dioxide. 3.fixed acidity and density.

The weekly correlated items are : 1.citric acid and volatile acidity. 2.fixed acidity and ph. 3.density and alcohol 4.ph and density

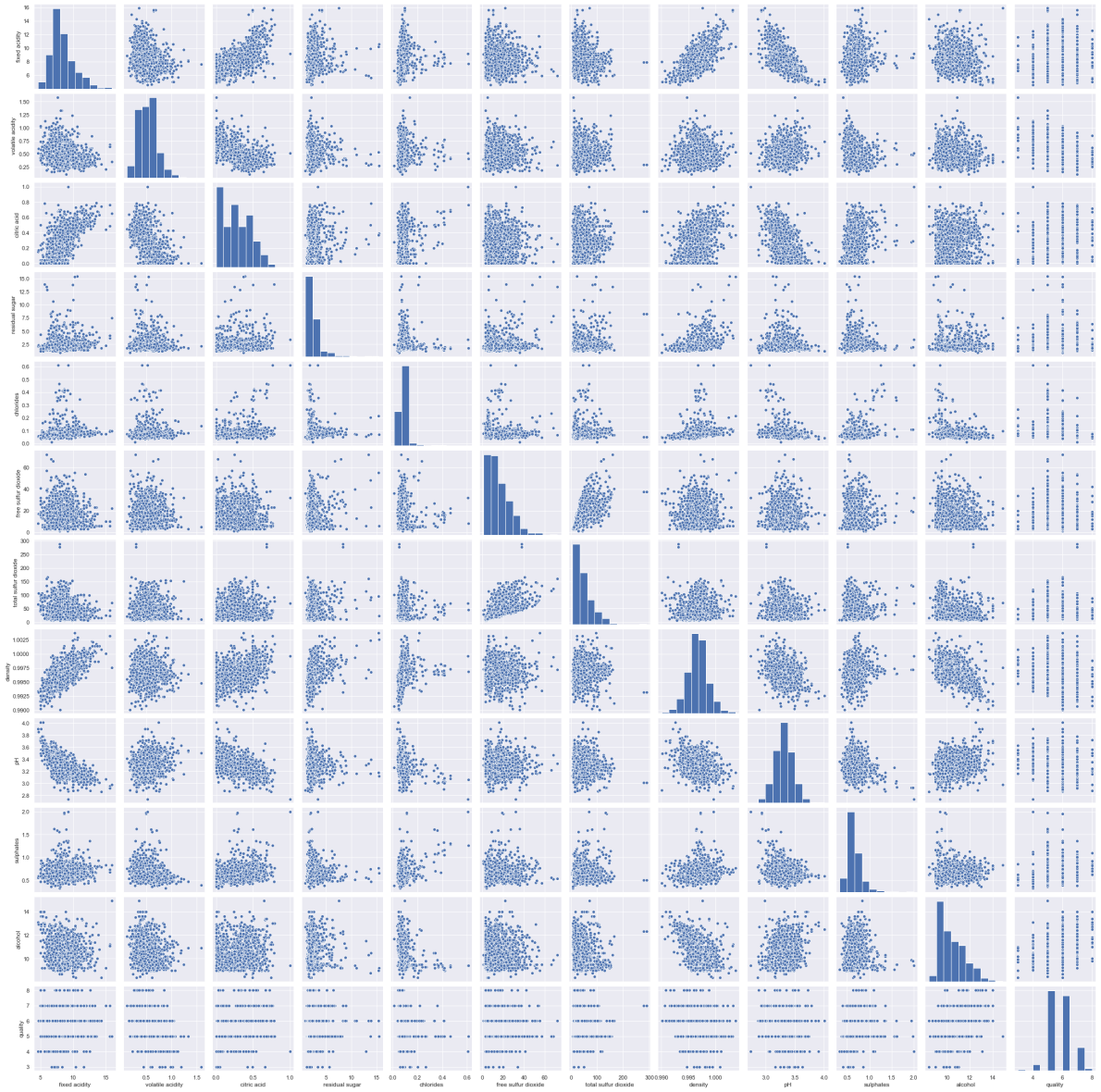
These are some relations which do not depend on each other at all.

In [14]:

```
sns.pairplot(data)
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0x2599de73b80>
```



In [15]:

```
data.columns
```

Out[15]:

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

Data preprocessing

In [16]:

```
#data['quality'] = data['quality'].map({3 : 'bad', 4 :'bad', 5: 'bad',
#                                       6: 'good', 7: 'good', 8: 'good'})
#I observed that the accurancy of model has increased when I considerd TV as 2 class classi
```

In [17]:

```
#data['quality'].value_counts()
```

In [18]:

```
# #from sklearn.preprocessing import LabelEncoder

# le = LabelEncoder()
# data['quality'].map({0:'bad',1:'good'})

# data['quality'] = le.fit_transform(data['quality'])

# data['quality'].value_counts
```

In [19]:

```
#sns.countplot(data['quality'])
```

In [20]:

```
# dividing the dataset into dependent and independent variables
X=data.iloc[:,:-1]
y=data.iloc[:,-1]
# determining the shape of x and y.
print(X.shape)
print(y.shape)
```

```
(1599, 11)
(1599,)
```

In [21]:

```
##Train Test Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.4,train_size=0.6,random_stat
```

In [22]:

```
## Standardize the dataset
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [23]:

```
X_train=scaler.fit_transform(X_train)
X_train
```

Out[23]:

```
array([[ 0.99064185, -0.87252436,  0.34803642, ..., -0.9151259 ,
        -0.04638563,  0.15582671],
       [ 1.04727864,  0.79403337,  0.60383572, ..., -0.9151259 ,
        -1.15693875, -0.39958949],
       [-0.65182509,  0.93291319, -0.72632064, ..., -0.13271365,
        -0.72505698, -0.30702012],
       ...,
       [-0.65182509,  0.23851413, -1.13559952, ...,  0.45409553,
        -0.35487261, -0.12188139],
       [-0.76509867,  0.90513722, -1.39139882, ...,  0.25849247,
        -0.35487261, -0.67729759],
       [ 0.93400506, -1.15028398,  1.62703292, ..., -0.71952283,
         0.75568051, -0.76986695]])
```

In [24]:

```
X_test=scaler.transform(X_test)
```

Modelling

Logistic Regression

In [25]:

```
from sklearn.linear_model import LogisticRegression
# creating the model
model = LogisticRegression(random_state=89)

# feeding the training set into the model
model.fit(X_train, y_train)

# predicting the results for the test set
y_pred = model.predict(X_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(X_train, y_train))
print("Testing accuracy :", model.score(X_test, y_test))

# classification report
print(classification_report(y_test, y_pred))

# confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
Training accuracy : 0.5922836287799792
Testing accuracy : 0.6265625
              precision    recall  f1-score   support

           3       1.00      0.50      0.67         2
           4       0.00      0.00      0.00        14
           5       0.70      0.76      0.73       281
           6       0.57      0.61      0.59       263
           7       0.52      0.33      0.41        75
           8       0.00      0.00      0.00         5

    accuracy                           0.63       640
   macro avg       0.47      0.37      0.40       640
weighted avg       0.61      0.63      0.61       640

[[  1   0   0   1   0   0]
 [  0   0   9   4   1   0]
 [  0   3 214  63   1   0]
 [  0   2  82 161  18   0]
 [  0   0   1  49  25   0]
 [  0   0   0   2   3   0]]

C:\Users\890234.CTS\Anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Decision Tree

In [26]:

```
from sklearn.tree import DecisionTreeClassifier

# creating model
model = DecisionTreeClassifier(random_state=42)

# feeding the training set into the model
model.fit(X_train, y_train)

# predicting the results for the test set
y_pred = model.predict(X_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(X_train, y_train))
print("Testing accuracy :", model.score(X_test, y_test))
```

```
Training accuracy : 1.0
Testing accuracy : 0.61875
```

In [27]:

```
# classification report
print(classification_report(y_test, y_pred))

# confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.06      0.14      0.09        14
           5       0.72      0.68      0.70       281
           6       0.63      0.61      0.62       263
           7       0.54      0.57      0.55        75
           8       0.00      0.00      0.00         5

    accuracy                           0.62       640
   macro avg       0.32      0.33      0.33       640
weighted avg       0.64      0.62      0.63       640

[[  0   0   1   1   0   0]
 [  1   2   7   4   0   0]
 [  0  16 191  68   6   0]
 [  1  12  57 160  28   5]
 [  0   3   8  20  43   1]
 [  0   0   0   2   3   0]]
```

In [28]:

```
#Now lets try to do some evaluation for decision tree model using cross validation.

model_eval = cross_val_score(estimator = model, X = X_train, y = y_train, cv = 10)
model_eval.mean()
```

```
C:\Users\890234.CTS\Anaconda3\lib\site-packages\sklearn\model_selection\_spl
it.py:670: UserWarning: The least populated class in y has only 8 members, w
hich is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

Out[28]:

```
0.5755921052631578
```

Random forest

In [29]:

```
from sklearn.ensemble import RandomForestClassifier

# creating the model
model = RandomForestClassifier(n_estimators = 200,random_state=6)

# feeding the training set into the model
model.fit(X_train, y_train)

# predicting the results for the test set
y_pred = model.predict(X_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(X_train, y_train))
print("Testing accuracy :", model.score(X_test, y_test))
```

```
Training accuracy : 1.0
Testing accuracy : 0.721875
```

In [30]:

```
model2 = RandomForestClassifier(random_state=64)
```

In [31]:

```
# Providing the different values of hyperparameters
param_dist = {'max_depth': [2, 3, 4, 8],
              'max_features': ['auto', 'sqrt', 'log2', None],
                'bootstrap' : [True, False],
              'criterion': ['gini', 'entropy']}

# Running gridsearchCV to check for all the different PnCs of these parameter values
cv_rf = GridSearchCV(model2, cv = 10,
                     param_grid=param_dist, verbose = True,
                     n_jobs = 3)

#Fitting the train set , so that grid search is executed on this dataset
cv_rf.fit(X_train, y_train)
```

```
Fitting 10 folds for each of 64 candidates, totalling 640 fits

C:\Users\890234.CTS\Anaconda3\lib\site-packages\sklearn\model_selection\_spl
it.py:670: UserWarning: The least populated class in y has only 8 members, w
hich is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done  44 tasks      | elapsed:   39.4s
[Parallel(n_jobs=3)]: Done 194 tasks      | elapsed:   56.5s
[Parallel(n_jobs=3)]: Done 444 tasks      | elapsed:  1.4min
[Parallel(n_jobs=3)]: Done 640 out of 640 | elapsed:  1.9min finished
```

Out[31]:

```
GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=64), n_job
s=3,
             param_grid={'bootstrap': [True, False],
                         'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 8],
                         'max_features': ['auto', 'sqrt', 'log2', None]},
             verbose=True)
```

In [32]:

```
#Printing the best parameters by using best_params
print('Best Parameters using grid search: \n', cv_rf.best_params_)
```

```
Best Parameters using grid search:
 {'bootstrap': True, 'criterion': 'entropy', 'max_depth': 8, 'max_features':
None}
```

In [33]:

```
#Finally the best parameters are specified

model2.set_params(criterion = 'gini',
                  max_features = 'auto',
                  bootstrap =  False,
                  max_depth = 8)
print("Training accuracy :", model.score(X_train, y_train))
print("Testing accuracy :", model.score(X_test, y_test))
```

```
Training accuracy : 1.0
Testing accuracy : 0.721875
```

In [34]:

```
model.get_params
```

Out[34]:

```
<bound method BaseEstimator.get_params of RandomForestClassifier(n_estimator
s=200, random_state=6)>
```

In [35]:

```
accuracy_rf = accuracy_score(y_test,y_pred)
accuracy_rf
```

Out[35]:

```
0.721875
```

In [36]:

```
# classification report
print(classification_report(y_test, y_pred))

# confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        14
           5       0.78      0.83      0.80       281
           6       0.69      0.71      0.70       263
           7       0.63      0.57      0.60        75
           8       0.00      0.00      0.00         5

    accuracy                           0.72       640
   macro avg       0.35      0.35      0.35       640
weighted avg       0.70      0.72      0.71       640

[[  0   1   1   0   0   0]
 [  0   0  10   4   0   0]
 [  0   1 233  47   0   0]
 [  0   1  54 186  22   0]
 [  0   2   1  29  43   0]
 [  0   0   0   2   3   0]]

C:\Users\890234.CTS\Anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [37]:

```
#Now lets try to do some evaluation for random forest model using cross validation.

model_eval = cross_val_score(estimator = model, X = X_train, y = y_train, cv = 10)
model_eval.mean()
```

```
C:\Users\890234.CTS\Anaconda3\lib\site-packages\sklearn\model_selection\_spl
it.py:670: UserWarning: The least populated class in y has only 8 members, w
hich is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

Out[37]:

```
0.6569736842105264
```

Support Vector Machine

In [38]:

```
from sklearn.svm import SVC

# creating the model
model = SVC(random_state=4)

# feeding the training set into the model
model.fit(X_train, y_train)

# predicting the results for the test set
y_pred = model.predict(X_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(X_train, y_train))
print("Testing accuracy :", model.score(X_test, y_test))
```

```
Training accuracy : 0.6611053180396246
Testing accuracy : 0.64375
```

In [39]:

```
# finding the best parameters for the SVC model

param = {
    'C': [0.8,0.9,1,1.1,1.2,1.3,1.4],
    'kernel':['linear', 'rbf'],
    'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4]
}
grid_svc = GridSearchCV(model, param_grid = param, scoring = 'accuracy', cv = 8)
```

In [40]:

```
grid_svc.fit(X_train, y_train)
```

Out[40]:

```
GridSearchCV(cv=8, estimator=SVC(random_state=4),
             param_grid={'C': [0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4],
                         'gamma': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4],
                         'kernel': ['linear', 'rbf']},
             scoring='accuracy')
```

In [41]:

```
grid_svc.best_params_
```

Out[41]:

```
{'C': 1.4, 'gamma': 1.3, 'kernel': 'rbf'}
```

In [42]:

```
# creating a new SVC model with these best parameters

model2 = SVC(C = 1.3, gamma = 0.8, kernel = 'rbf')
model2.fit(X_train, y_train)
y_pred = model2.predict(X_test)

print(classification_report(y_test, y_pred))
print("Training accuracy :", model2.score(X_train, y_train))
print("Testing accuracy :", model2.score(X_test, y_test))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        14
           5       0.70      0.80      0.75       281
           6       0.65      0.67      0.66       263
           7       0.78      0.48      0.60        75
           8       0.00      0.00      0.00         5

    accuracy                           0.68       640
   macro avg       0.35      0.32      0.33       640
weighted avg       0.66      0.68      0.67       640

Training accuracy : 0.9520333680917622
Testing accuracy : 0.68125

C:\Users\890234.CTS\Anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

```
```

In [ ]:

```
```