In [27]:
```
# import libraries
import pandas as pd
import numpy as np
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
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, cro
ss_val_score
%matplotlib inline
```

In [2]:
```
data = pd.read_csv("redwine.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [4]:
```
data.columns
```

Out[4]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual su
```

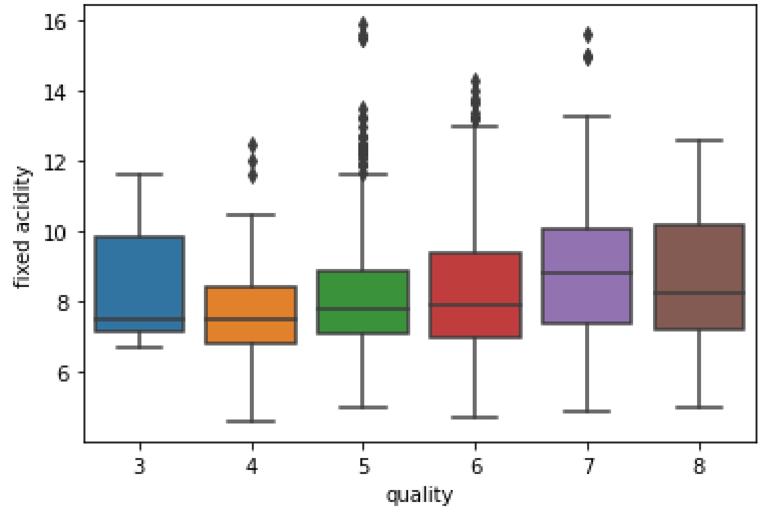
```
gar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'den
sity',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [5]:
```
data.info()
```

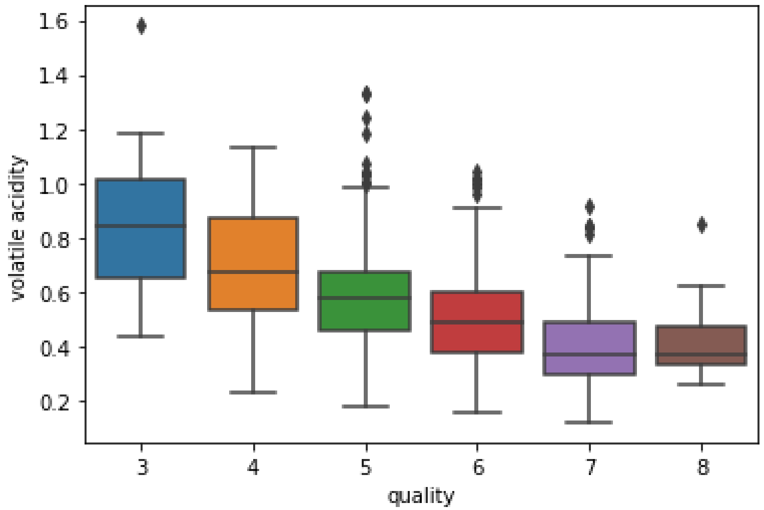
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   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 [6]:
```
sns.boxplot('quality', 'fixed acidity', data = data)
```

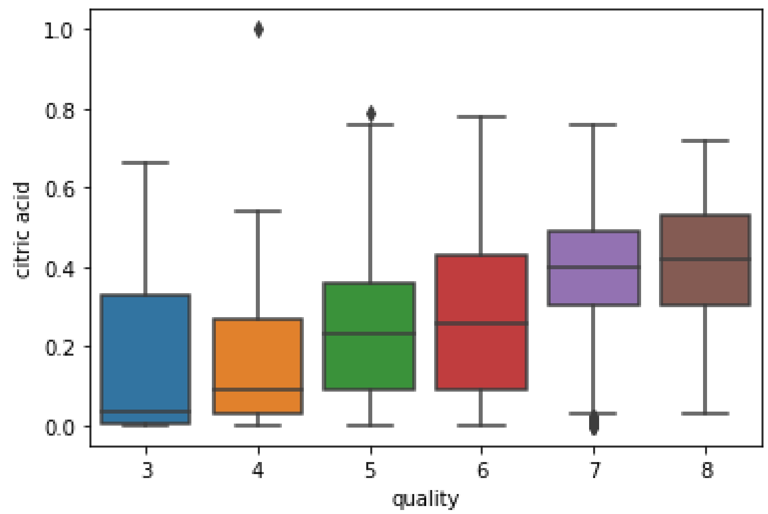
Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x231d08eb550>
```



In [7]:
```
sns.boxplot('quality', 'volatile acidity', data = data)
```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x231d1024c40>

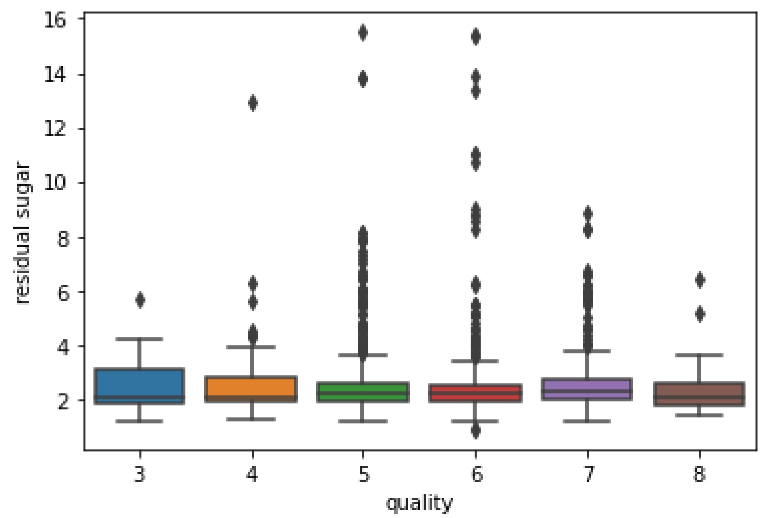


In [8]:
```
sns.boxplot('quality', 'citric acid', data = data)
```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x231d11792e0>



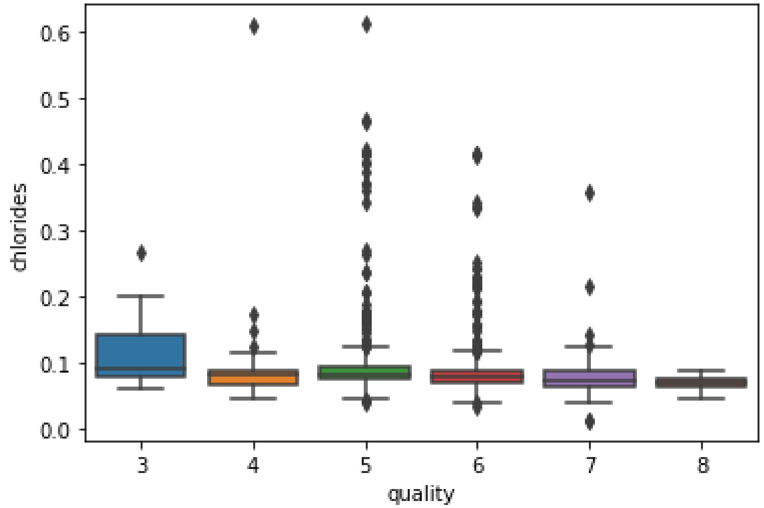
In [9]:
```
sns.boxplot('quality', 'residual sugar', data = data)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x231d11791c0>



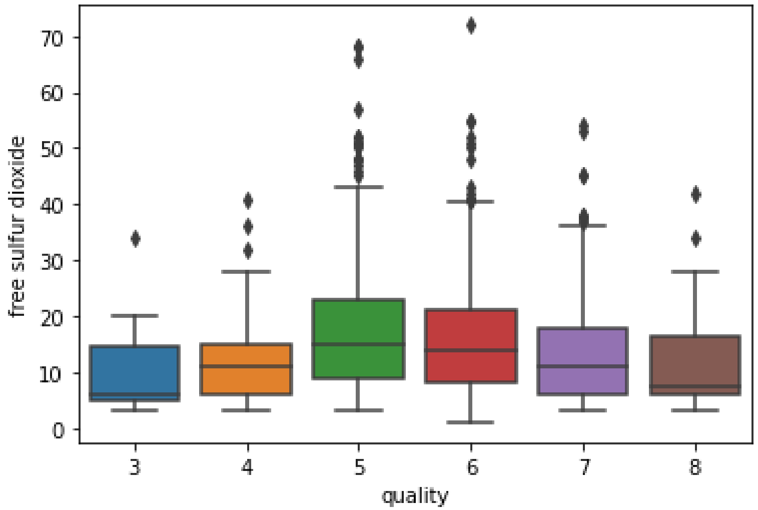
In [10]:
```
sns.boxplot('quality', 'chlorides', data = data)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x231d13025b0>



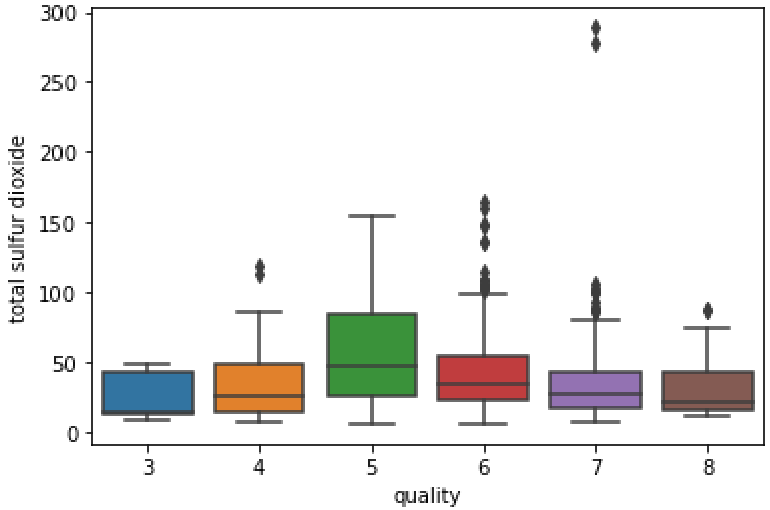
In [11]:
```
sns.boxplot('quality', 'free sulfur dioxide', data = data)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x231d13bf400>



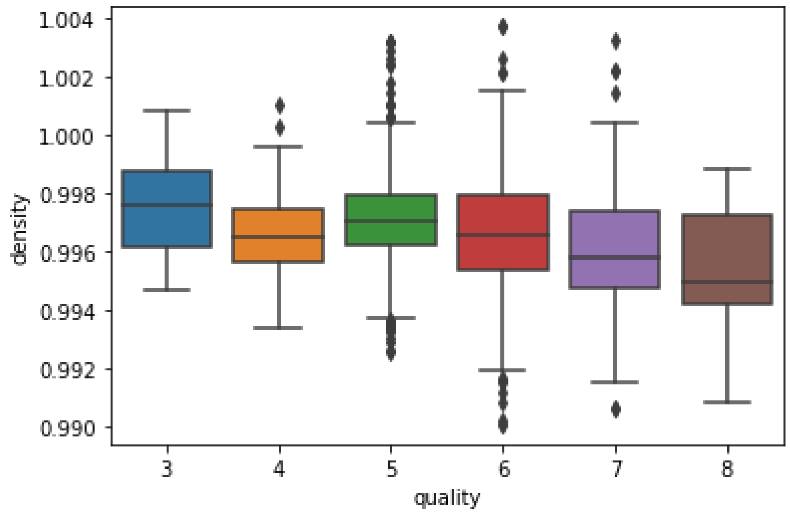
In [12]:
```
sns.boxplot('quality', 'total sulfur dioxide', data = data)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x231d1474820>



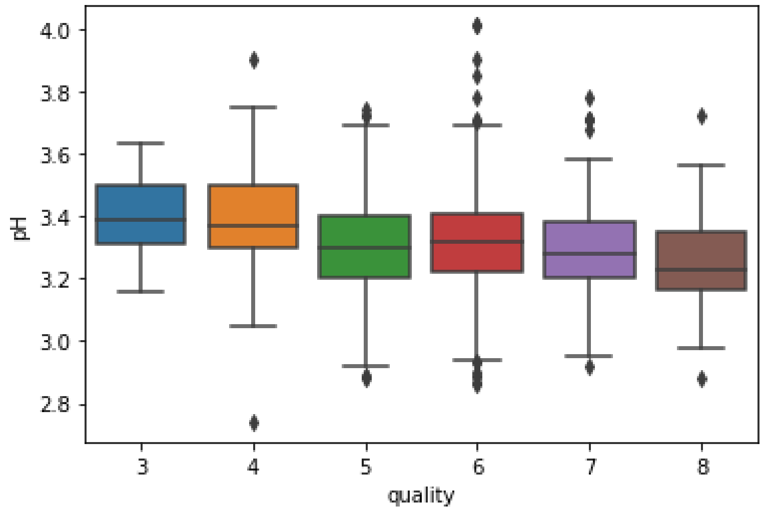
In [13]:
```
sns.boxplot('quality', 'density', data = data)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x231d1523280>



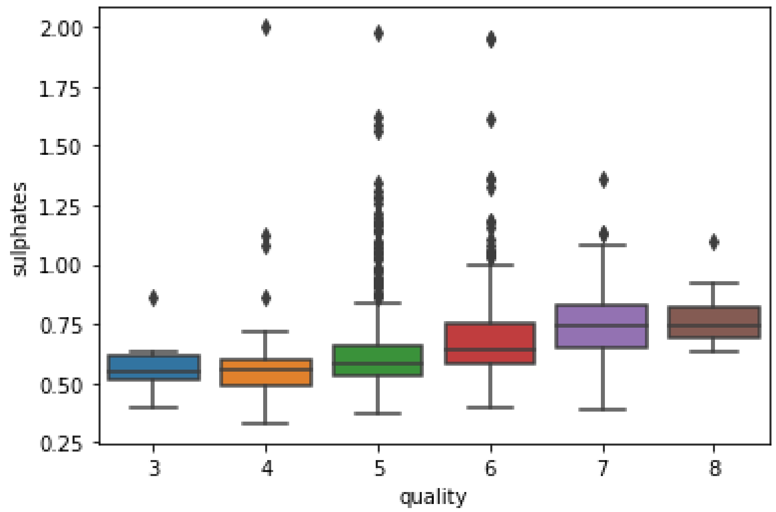
In [14]:
```python
sns.boxplot('quality', 'pH', data = data)
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x231d15fbfd0>



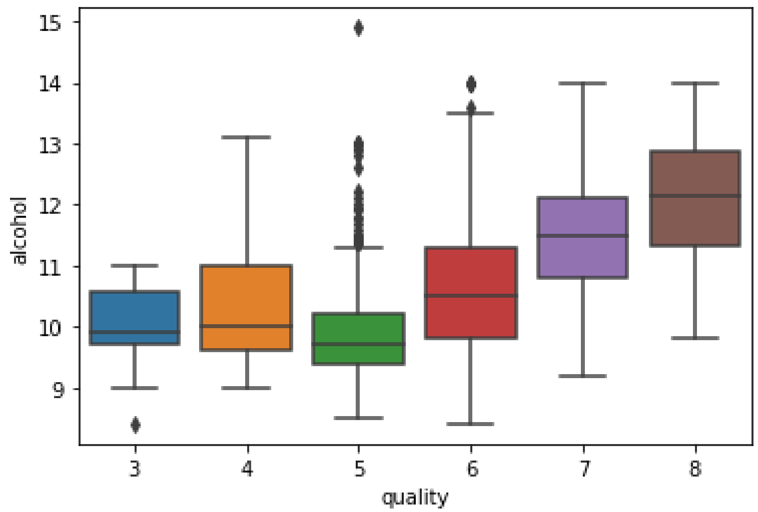
In [15]:
```python
sns.boxplot('quality', 'sulphates', data = data)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x231d16ab460>



In [16]:
```
sns.boxplot('quality', 'alcohol', data = data)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x231d140f6d0>



In [17]:
```
data.describe()
```

Out[17]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulf dioxic
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0000(
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4677(
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.8953:
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0000(
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.0000(
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.0000(
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.0000(
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.0000(

In [24]:
```
bins = (2, 6.5, 8)
group_names = ['bad', 'good']
data['quality'] = pd.cut(data['quality'], bins = bins, labels = group_n
ames)
```

In [28]:
```
label_quality = LabelEncoder()
```

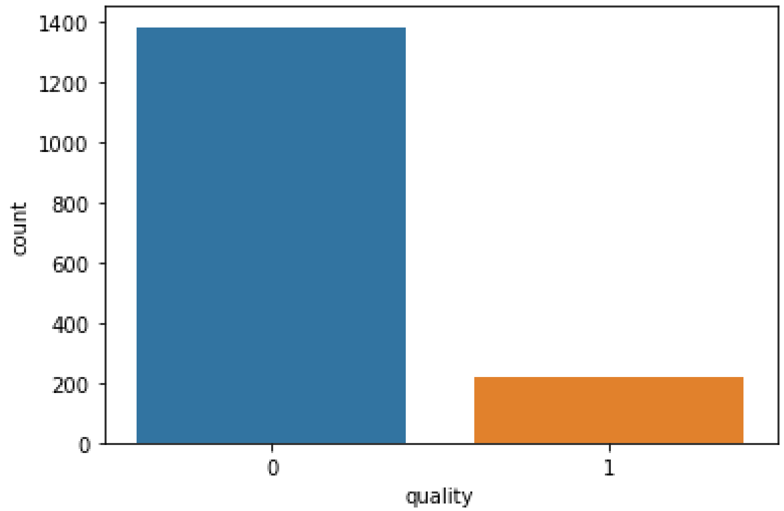
In [31]:
```
data['quality'] = label_quality.fit_transform(data['quality'])
```

In [33]:
```
data['quality'].value_counts()
```

Out[33]:
```
0    1382
1     217
Name: quality, dtype: int64
```

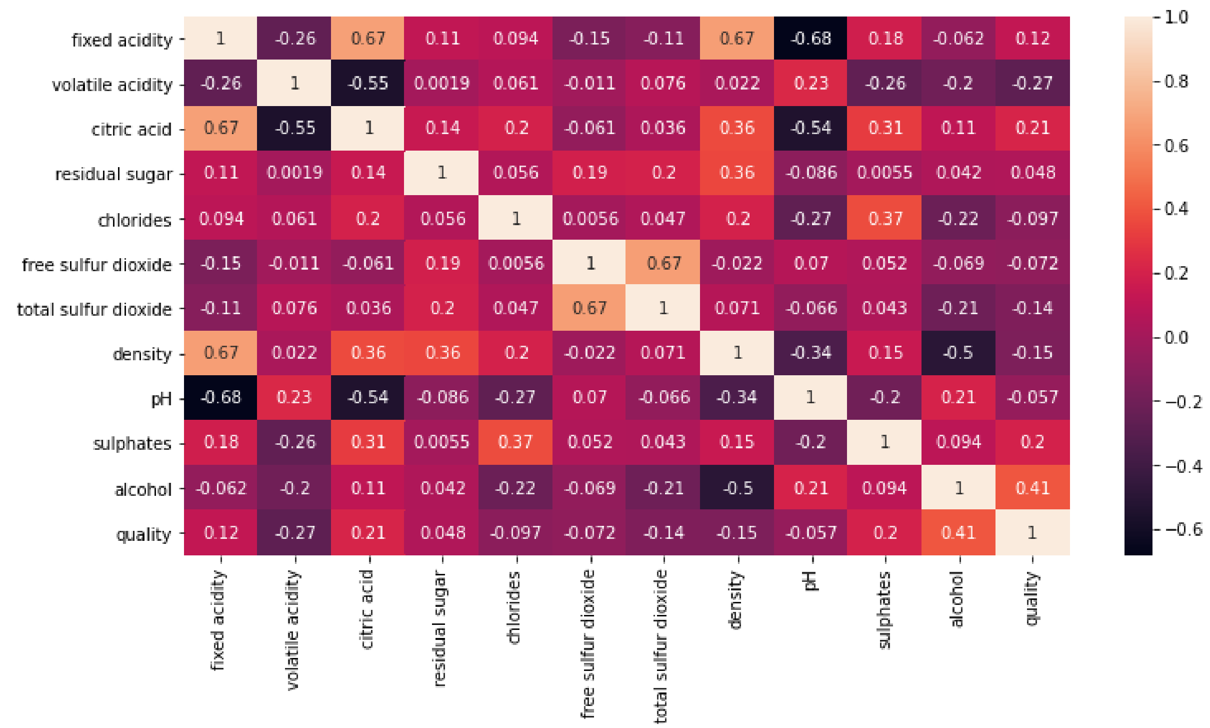
In [35]:
```
sns.countplot(data['quality'])
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x231d1ee6820>
```



In [38]:
```
plt.figure(figsize=(12,6))
sns.heatmap(data.corr(),annot=True)
```

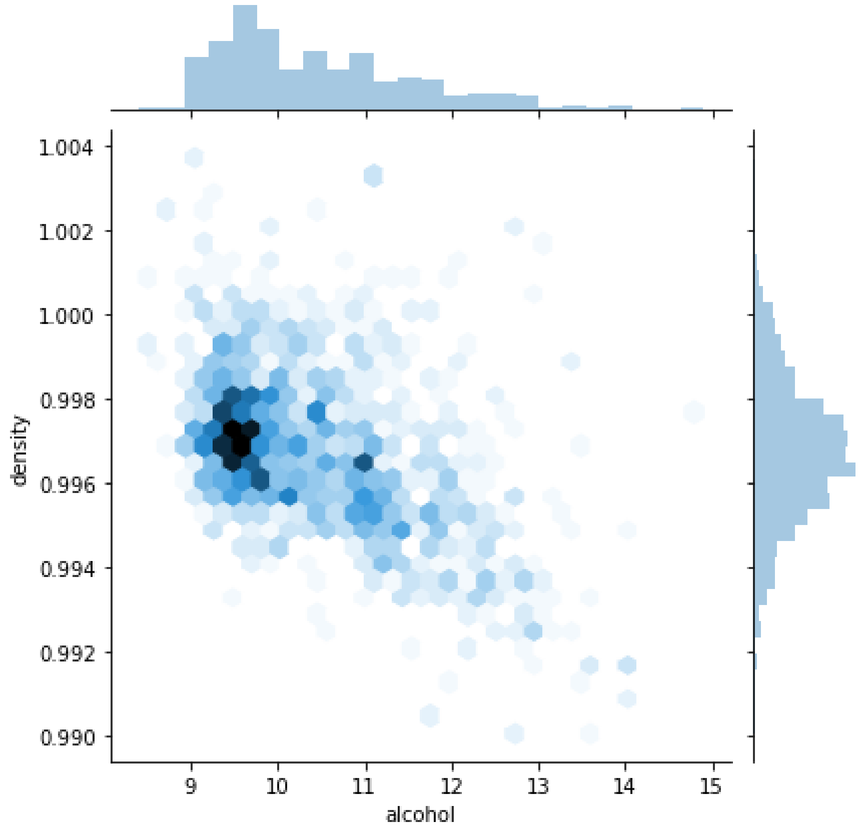
```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x231d1b2de20>
```




```
In [40]: plt.figure(figsize=(12,6))
         sns.jointplot(y=data["density"],x=data["alcohol"],kind="hex")
```

```
Out[40]: <seaborn.axisgrid.JointGrid at 0x231d2325460>

         <Figure size 864x432 with 0 Axes>
```



In [44]:
```
X = data.drop('quality', axis = 1)
y = data['quality']
```

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 42)
```

In [46]:
```
sc = StandardScaler()
```

In [47]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

In [58]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_predict=rf.predict(X_test)
```

In [59]:
```
rf_conf_matrix = confusion_matrix(y_test, rf_predict)
rf_acc_score = accuracy_score(y_test, rf_predict)
print(rf_conf_matrix)
print(rf_acc_score*100)
```

```
[[264   9]
 [ 28  19]]
88.4375
```

In [53]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_predict = lr.predict(X_test)
```

In [54]:
```
lr_conf_matrix = confusion_matrix(y_test, lr_predict)
lr_acc_score = accuracy_score(y_test, lr_predict)
print(lr_conf_matrix)
print(lr_acc_score*100)
```

```
[[268   5]
 [ 35  12]]
87.5
```

In [55]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
dt_predict = dt.predict(X_test)
```

In [56]:
```
dt_conf_matrix = confusion_matrix(y_test, dt_predict)
dt_acc_score = accuracy_score(y_test, dt_predict)
```

```
print(dt_conf_matrix)
print(dt_acc_score*100)
```

```
[[247  26]
 [ 24  23]]
84.375
```

In [65]:
```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train,y_train)
pred_svc =svc.predict(X_test)
```

In [66]:
```
from sklearn.metrics import classification_report,accuracy_score
print(classification_report(y_test,pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       273
           1       0.71      0.26      0.37        47

    accuracy                           0.88       320
   macro avg       0.80      0.62      0.65       320
weighted avg       0.86      0.88      0.85       320
```

In [63]:
```
lin_svc_conf_matrix = confusion_matrix(y_test, rf_predict)
lin_svc_acc_score = accuracy_score(y_test, rf_predict)
print(lin_svc_conf_matrix)
print(lin_svc_acc_score*100)
```

```
[[264   9]
 [ 28  19]]
88.4375
```

In [68]:
```
conclusion = pd.DataFrame({'models': ["Random Forest","Logistic Regress
ion","Decission Tree","Supprot vector machine"],
                           'accuracies': [accuracy_score(y_test, rf_pre
dict),accuracy_score(y_test, lr_predict),accuracy_score(y_test, dt_pred
```

```
ict),accuracy_score(y_test,pred_svc)]})
conclusion
```

Out[68]:

	models	accuracies
0	Random Forest	0.884375
1	Logistic Regression	0.875000
2	Decission Tree	0.843750
3	Supprot vector machine	0.875000

In [ ]:

```
```