In [28]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
%matplotlib inline
```

Explore and Clean the Data

In [2]:

```
os.getcwd()
```

Out[2]:

```
'C:\\Users\\tural'
```

In [3]:

```
os.chdir("C:\\Users\\tural\\OneDrive\\Desktop\\Study Materials\\Datasets")
```

In [4]:

```
df = pd.read_csv("WineQT.csv")
df.head(20)
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	c
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
5	7.4	0.660	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	
6	7.9	0.600	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	
7	7.3	0.650	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	
8	7.8	0.580	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	
9	6.7	0.580	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	9.2	
10	5.6	0.615	0.00	1.6	0.089	16.0	59.0	0.9943	3.58	0.52	9.9	
11	7.8	0.610	0.29	1.6	0.114	9.0	29.0	0.9974	3.26	1.56	9.1	
12	8.5	0.280	0.56	1.8	0.092	35.0	103.0	0.9969	3.30	0.75	10.5	
13	7.9	0.320	0.51	1.8	0.341	17.0	56.0	0.9969	3.04	1.08	9.2	
14	7.6	0.390	0.31	2.3	0.082	23.0	71.0	0.9982	3.52	0.65	9.7	
15	7.9	0.430	0.21	1.6	0.106	10.0	37.0	0.9966	3.17	0.91	9.5	
16	8.5	0.490	0.11	2.3	0.084	9.0	67.0	0.9968	3.17	0.53	9.4	
17	6.9	0.400	0.14	2.4	0.085	21.0	40.0	0.9968	3.43	0.63	9.7	
18	6.3	0.390	0.16	1.4	0.080	11.0	23.0	0.9955	3.34	0.56	9.3	
19	7.6	0.410	0.24	1.8	0.080	4.0	11.0	0.9962	3.28	0.59	9.5	

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [6]:

```
a = np.arange(1143)
df['Id'] = a
df.columns = df.columns.str.lower()
df.columns = df.columns.str.replace(' ', '_')
df.columns = df.columns.str.replace('_dioxide', '')
df.round(3)
```

Out[6]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur	total_sulfur	der
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0
...	...	...	...	...	...	...	...	
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0

1143 rows × 13 columns

In [ ]:

Descriptive Statistics

In [8]:

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

Out[8]:

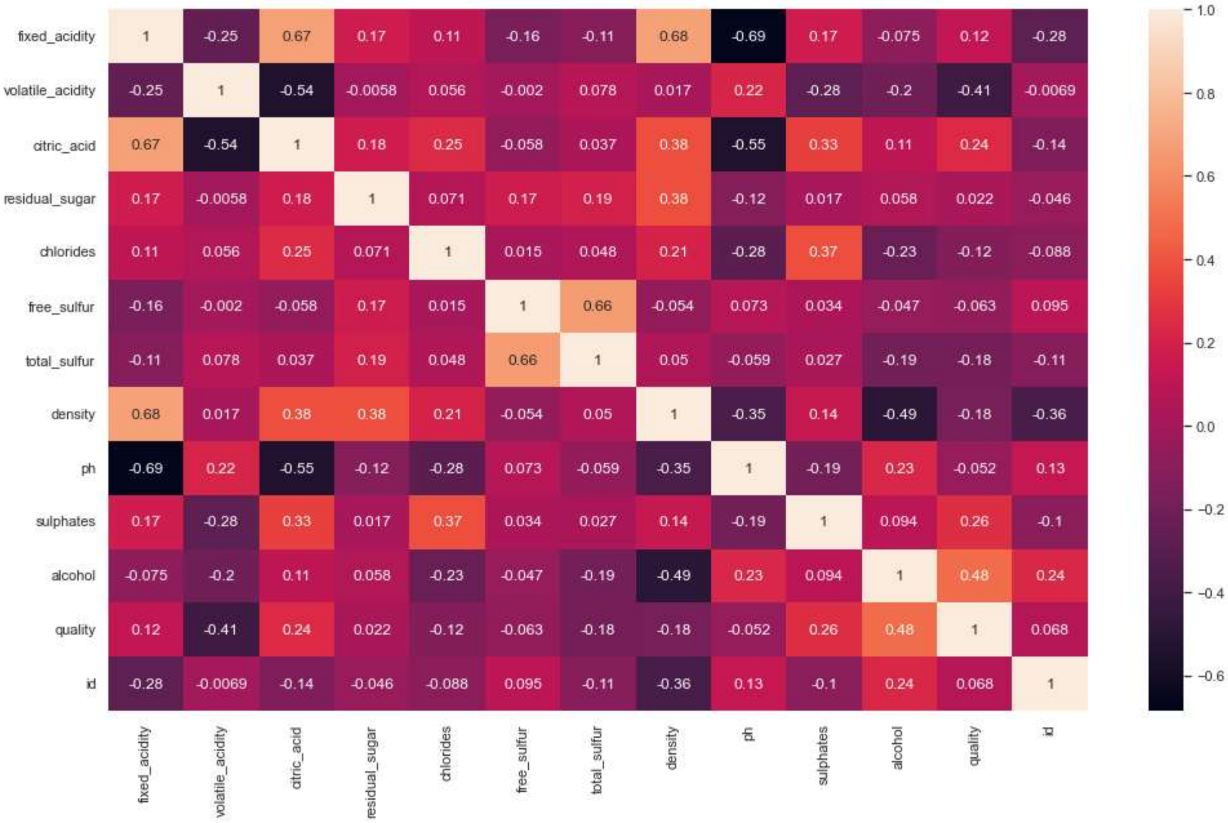
```
quality
5    483
6    462
7    143
4     33
8     16
3      6
dtype: int64
```

In [9]:

```
sns.set(rc = {'figure.figsize':(17,10)})
sns.heatmap(df.corr(), annot = True)
```

Out[9]:

```
<AxesSubplot:>
```



Lets create groups for quality and divide it into 2 groups which are poor quality(0) and good quality(1)

In [11]:

```
df['label'] = pd.cut(x=df['quality'], bins=[0, 5, 8,],labels=[0, 1])
```

Decision Tree Model

As we see there is not any highly correlated variable to quality. That is the reason we cannot use linear regression and we will be choosing decision tree model.

In [14]:

```
df.set_index('id')
```

Out[14]:

id	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur	total_sulfur	de
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9
...	...	...	...	...	...	...	...	
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.9
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.9
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.9
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.9
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.9

1143 rows × 13 columns

In [21]:
```
X = df.drop('label', axis = 1)
```

In [22]:
```
y = df['label']
```

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [33]:
```
dtree = DecisionTreeClassifier()
```

In [34]:
```
dtree.fit(X_train,y_train)
```

Out[34]:
```
DecisionTreeClassifier()
```

In [35]:
```
predictions = dtree.predict(X_test)
```

In [36]:
```
print(confusion_matrix(y_test,predictions))
print("\n")
print(classification_report(y_test, predictions))
```

```
[[155   0]
 [  0 188]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00       155
           1       1.00      1.00      1.00       188

    accuracy                           1.00       343
   macro avg       1.00      1.00      1.00       343
weighted avg       1.00      1.00      1.00       343
```

In [39]:

```
rfc = RandomForestClassifier(n_estimators = 200)
```

In [40]:

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

Out[40]:

```
RandomForestClassifier(n_estimators=200)
```

In [41]:

```
rfc_pred = rfc.predict(X_test)
```

In [42]:

```
print(confusion_matrix(y_test,rfc_pred))
print("\n")
print(classification_report(y_test, rfc_pred))
```

```
[[155   0]
 [  0 188]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00       155
           1       1.00      1.00      1.00       188

    accuracy                           1.00       343
   macro avg       1.00      1.00      1.00       343
weighted avg       1.00      1.00      1.00       343
```

As we can see from the confusion metrics, decision tree works perfectly fine and there is not any type of errors. We can conclude that we do not need to create random forests for this dataset and we can predict the wine quality with decisiion trees.