In [134]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [135]:
```
df = pd.read_csv("winequality-red.csv")
```

In [136]:
```
df.head()
```

Out[136]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoho
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 [137]:
```
df.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 [138]:
```
df.columns.tolist()
```

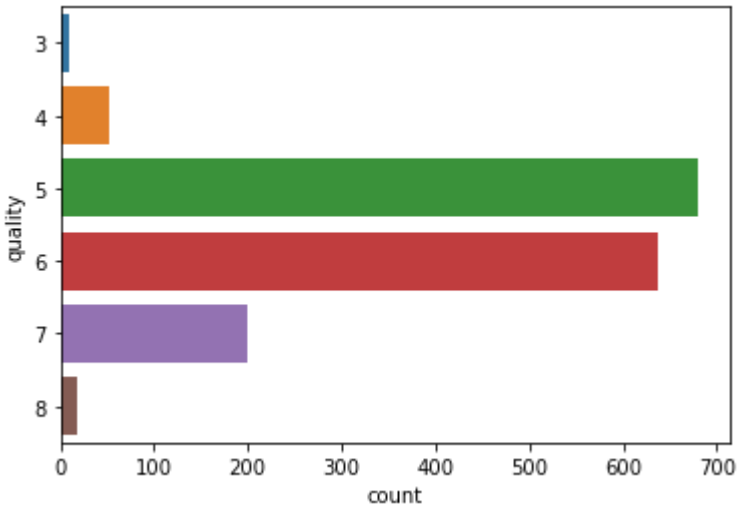
Out[138]:
```
['fixed acidity',
 'volatile acidity',
 'citric acid',
 'residual sugar',
 'chlorides',
 'free sulfur dioxide',
 'total sulfur dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol',
 'quality']
```

In [139]:
```
vc=df["quality"].value_counts()
vc
```

Out[139]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

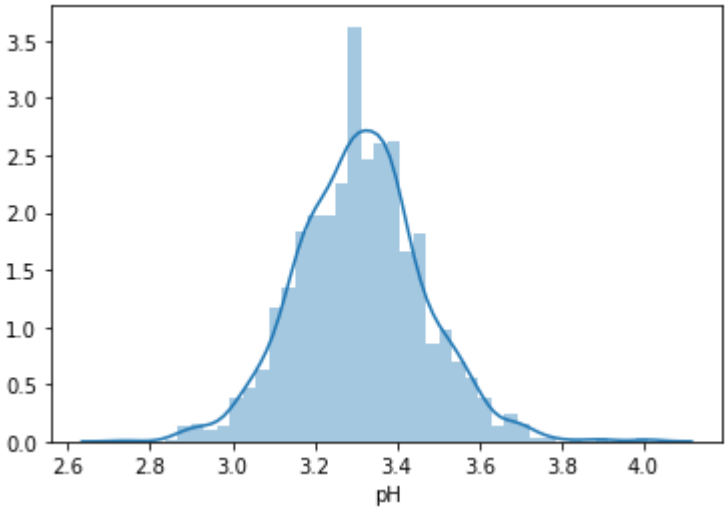
In [140]:
```
sns.countplot(data=df,y="quality")
```

Out[140]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1eb91818c88>
```



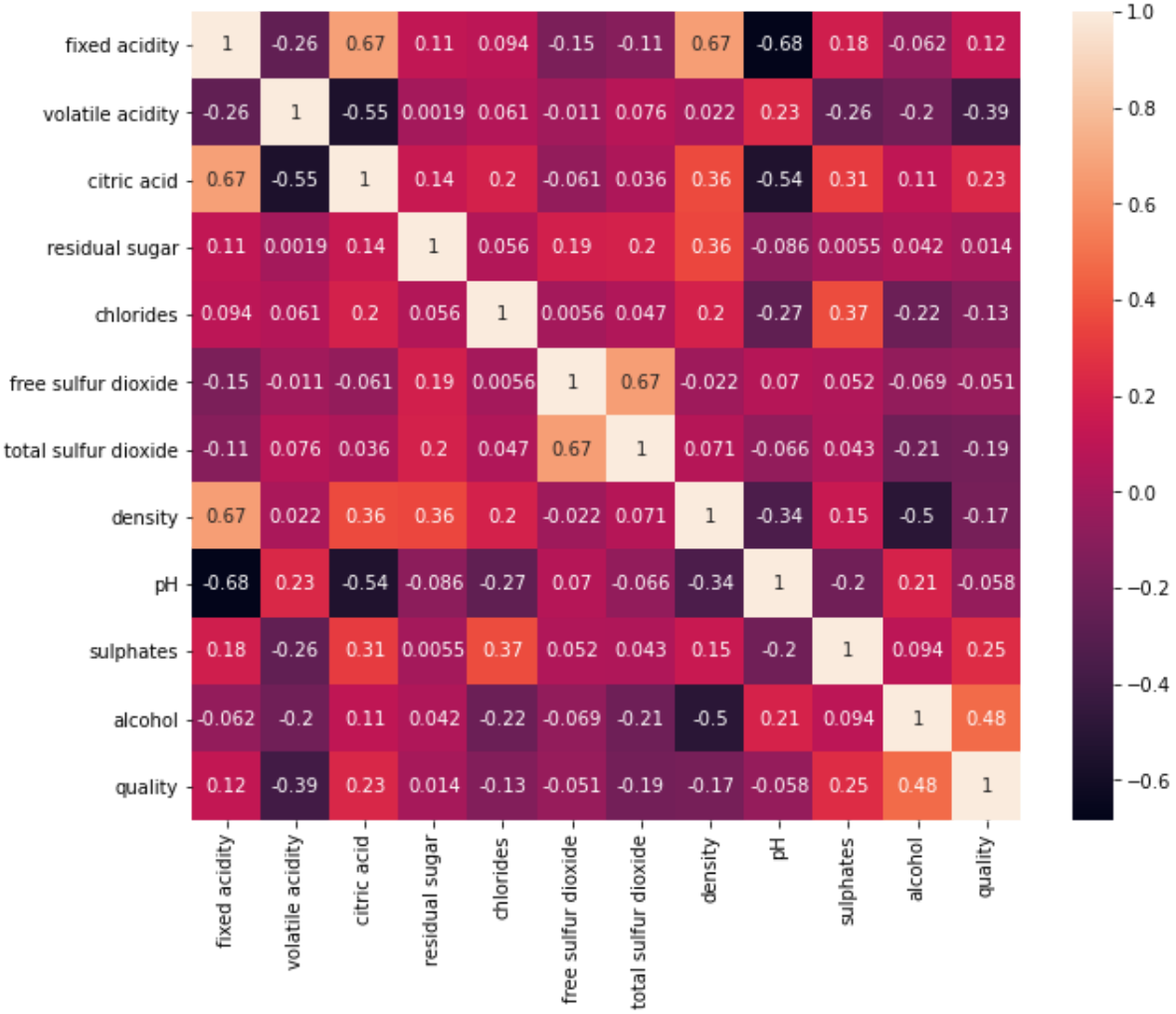
In [141]:
```
sns.distplot(df["pH"])  # pH distribution
```

Out[141]: <matplotlib.axes._subplots.AxesSubplot at 0x1eb91932548>



In [142]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
```

Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x1eb919874c8>



In [143]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
```

In [144]:
```
df.isnull().sum()
```

Out[144]:
```
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
```

In [145]:
```
## preparing data for clustering
```

In [146]:
```
target_col = df["quality"]
```

In [147]:
```
target_col
```

Out[147]:
```
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
```

In [148]:
```
sc = StandardScaler()
```

In [149]:
```
df_scaled = sc.fit_transform(df)
```

In [150]:
```
df_scaled
```

Out[150]:
```
array([[-0.52835961,  0.96187667, -1.39147228, ..., -0.57920652,
        -0.96024611, -0.78782264],
       [-0.29854743,  1.96744245, -1.39147228, ...,  0.1289504 ,
        -0.58477711, -0.78782264],
       [-0.29854743,  1.29706527, -1.18607043, ..., -0.04808883,
        -0.58477711, -0.78782264],
       ...,
       [-1.1603431 , -0.09955388, -0.72391627, ...,  0.54204194,
         0.54162988,  0.45084835],
       [-1.39015528,  0.65462046, -0.77526673, ...,  0.30598963,
        -0.20930812, -0.78782264],
       [-1.33270223, -1.21684919,  1.02199944, ...,  0.01092425,
         0.54162988,  0.45084835]])
```

In [151]:
```
df_scaled = pd.DataFrame(df_scaled,columns=df.columns)
df_scaled
```

Out[151]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	-0.466193	-0.379133	0.558274	1.28
1	-0.298547	1.967442	-1.391472	0.043416	0.223875	0.872638	0.624363	0.028261	-0.7
2	-0.298547	1.297065	-1.186070	-0.169427	0.096353	-0.083669	0.229047	0.134264	-0.3
3	1.654856	-1.384443	1.484154	-0.453218	-0.264960	0.107592	0.411500	0.664277	-0.9
4	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	-0.466193	-0.379133	0.558274	1.28
...	...	...	...	...	...	...	...	...	
1594	-1.217796	0.403229	-0.980669	-0.382271	0.053845	1.542054	-0.075043	-0.978765	0.8
1595	-1.390155	0.123905	-0.877968	-0.240375	-0.541259	2.211469	0.137820	-0.862162	1.3
1596	-1.160343	-0.099554	-0.723916	-0.169427	-0.243707	1.255161	-0.196679	-0.533554	0.7
1597	-1.390155	0.654620	-0.775267	-0.382271	-0.264960	1.542054	-0.075043	-0.676657	1.6
1598	-1.332702	-1.216849	1.021999	0.752894	-0.434990	0.203223	-0.135861	-0.666057	0.5

1599 rows × 12 columns

In [152]:
```
kmeans = KMeans(
        n_clusters=6,
        max_iter=1000,
        random_state=1,
        init="random"
            )
```

In [153]:
```
kmeans.fit(df_scaled)
```

Out[153]:
```
KMeans(init='random', max_iter=1000, n_clusters=6, random_state=1)
```

In [154]:
```
y_km = kmeans.predict(df_scaled)
```

In [159]:
```
cluster_centre  = pd.DataFrame(kmeans.cluster_centers_,columns=df.columns)
```

In [160]:
```
cluster_centre
```

Out[160]:

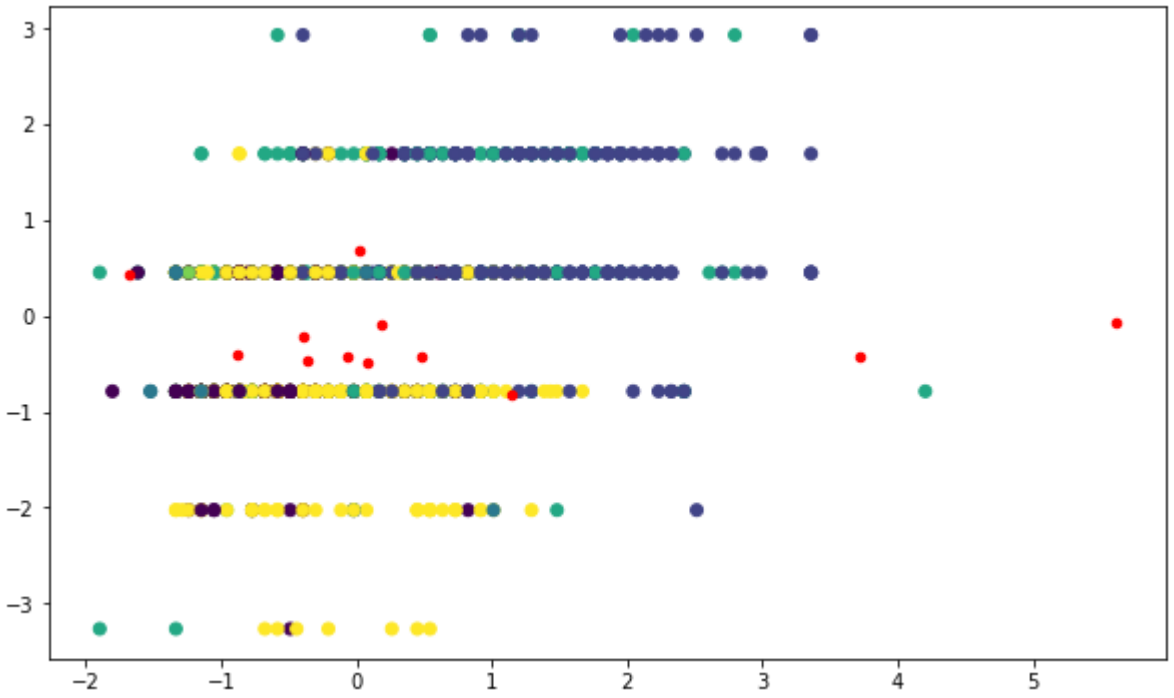
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	p
0	-0.041842	0.072370	0.069753	-0.100205	-0.030330	0.937458	1.187992	0.264947	-0.1427(
1	-0.618848	-0.467039	-0.125523	-0.226296	-0.384579	0.143304	-0.231554	-1.158374	0.5410(
2	-0.188637	-0.051583	0.400212	4.244759	0.206318	1.589869	1.742223	1.035402	-0.1945(
3	1.378218	-0.689121	1.153195	0.102163	-0.004633	-0.565416	-0.546030	0.799599	-0.8560(
4	0.081831	0.017955	1.144178	-0.399396	5.604731	-0.070479	0.474416	0.185803	-1.6873(
5	-0.489153	0.691740	-0.819240	-0.214826	-0.070664	-0.434285	-0.419141	-0.087270	0.4307(

In [161]:
```
df.columns
```

Out[161]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [179]:
```
plt.figure(figsize=(10,6))
plt.scatter(df_scaled["alcohol"],df_scaled["quality"],c=y_km)
plt.scatter(kmeans.cluster_centers_[-2],kmeans.cluster_centers_[-1],c="red",s=
20)
```

Out[179]: <matplotlib.collections.PathCollection at 0x1eb9343be48>

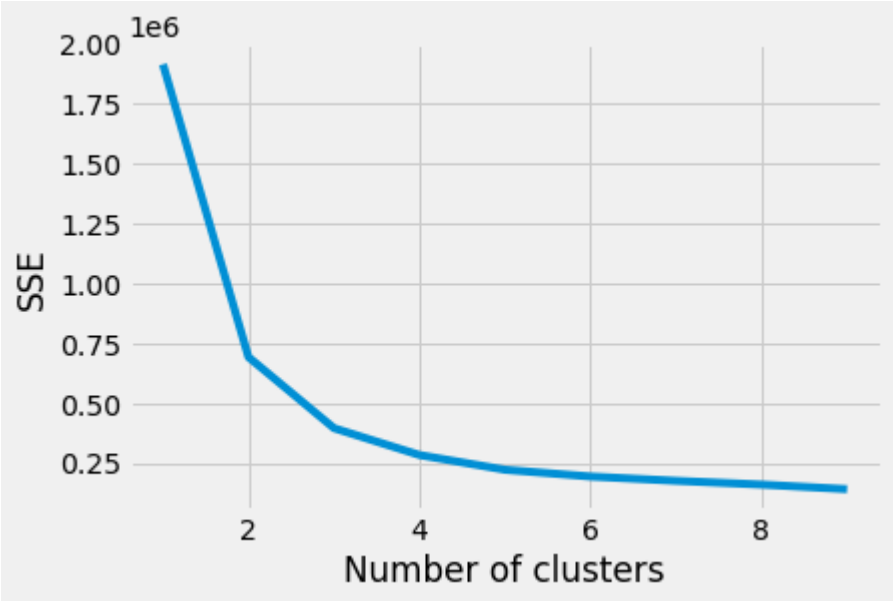


In [186]:
```
!pip install kneed
from kneed import KneeLocator
```

```
Requirement already satisfied: kneed in c:\users\dhruv\anaconda3\envs\ml\lib
\site-packages (0.7.0)
Requirement already satisfied: matplotlib in c:\users\dhruv\anaconda3\envs\ml
\lib\site-packages (from kneed) (3.2.2)
Requirement already satisfied: scipy in c:\users\dhruv\anaconda3\envs\ml\lib
\site-packages (from kneed) (1.5.2)
Requirement already satisfied: numpy>=1.14.2 in c:\users\dhruv\anaconda3\envs
\ml\lib\site-packages (from kneed) (1.19.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dhruv\anaconda3
\envs\ml\lib\site-packages (from matplotlib->kneed) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\dhruv\anaconda3\envs
\ml\lib\site-packages (from matplotlib->kneed) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\dhruv\anaconda3\envs\ml\lib\site-packages (from matplotlib->kneed)
(2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\dhruv\anacond
a3\envs\ml\lib\site-packages (from matplotlib->kneed) (2.8.1)
Requirement already satisfied: six in c:\users\dhruv\anaconda3\envs\ml\lib\si
te-packages (from cycler>=0.10->matplotlib->kneed) (1.15.0)
```

In [187]:
```
kmeans_kwargs = {"init": "random", "n_init": 10, "max_iter": 300, "random_state": 42}

sse = []
for i in range(1,10,1):
    kmeans = KMeans(n_clusters=i, **kmeans_kwargs).fit(df)
    sse.append(kmeans.inertia_)
plt.style.use("fivethirtyeight")
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.plot(range(1,10,1), sse)
plt.show()
kl = KneeLocator(range(1, 10, 1), sse, curve="convex", direction="decreasing")
print(kl.elbow)
```




```
3
```

In [ ]:
```
## thus we successfully performed KMEANS clustering on Red-wine quality dataset.
```