In [21]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```python
Data = pd.read_csv(r"C:\Users\santh\Downloads\archive (6)\winequality-red.cs
Data.head()
```

Out[3]:

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

In [4]:
```python
Data.shape
```

Out[4]:
```
(1599, 12)
```

In [5]:
```python
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 [7]:
```
Data.describe()
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total s di
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0(
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4(
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.8(
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0(
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.0(
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.0(
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.0(
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.0(

In [8]:
```
Data.duplicated().sum()
```

Out[8]: 240

In [10]:
```
df = Data[~Data.duplicated()]
df.shape
```

Out[10]: (1359, 12)

In [11]:
```
df.duplicated().sum()
```

Out[11]: 0

In [12]:
```
df.describe()
```

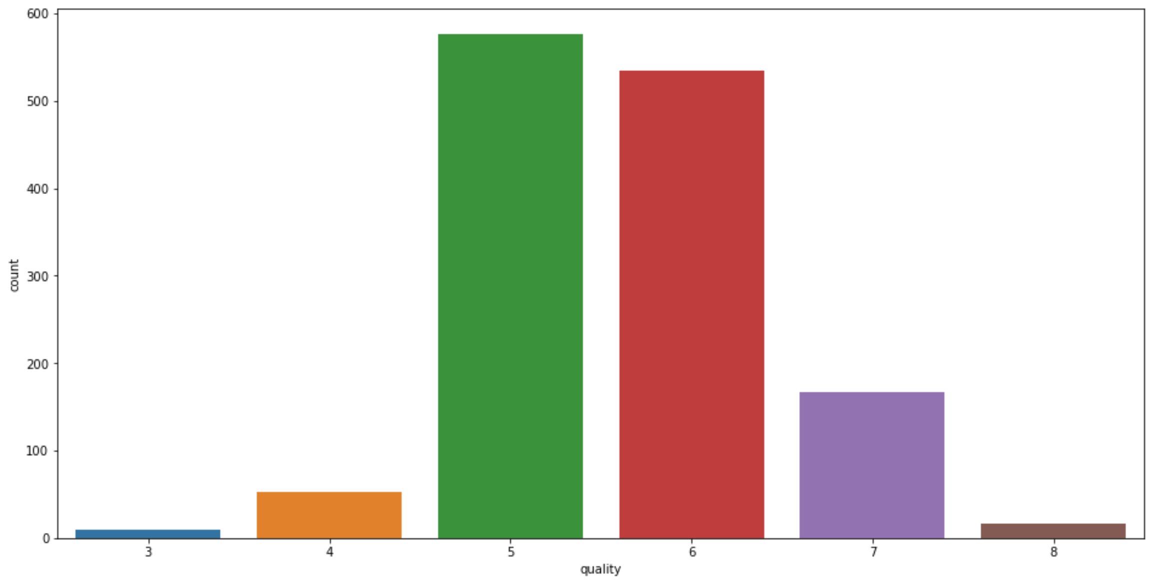
Out[12]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total s di
count	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000	1359.0(
mean	8.310596	0.529478	0.272333	2.523400	0.088124	15.893304	46.8(
std	1.736990	0.183031	0.195537	1.352314	0.049377	10.447270	33.4(
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0(
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.0(
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.0(
75%	9.200000	0.640000	0.430000	2.600000	0.091000	21.000000	63.0(
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.0(

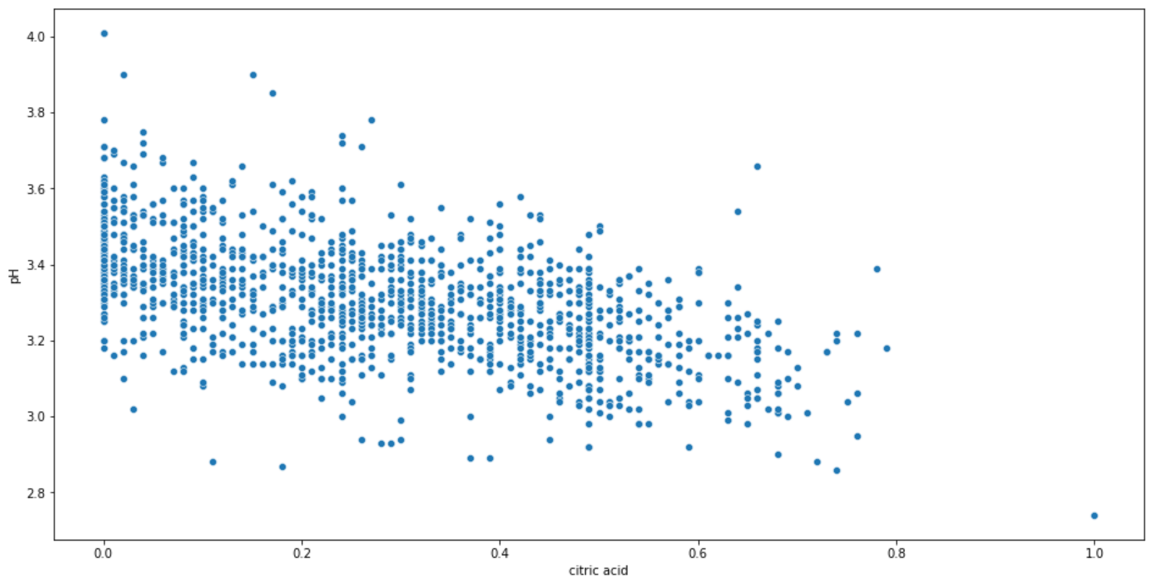
In [13]:
```
from sklearn.model_selection import train_test_split
```

Identifying the Minority class

In [76]:
```
plt.figure(figsize=(16,8))
sns.countplot(df["quality"], data=df)
plt.show()
```



In [24]:
```
plt.figure(figsize=(16,8))
sns.scatterplot(x=df["citric acid"],y=df["pH"] ,data=df)
plt.show()
```



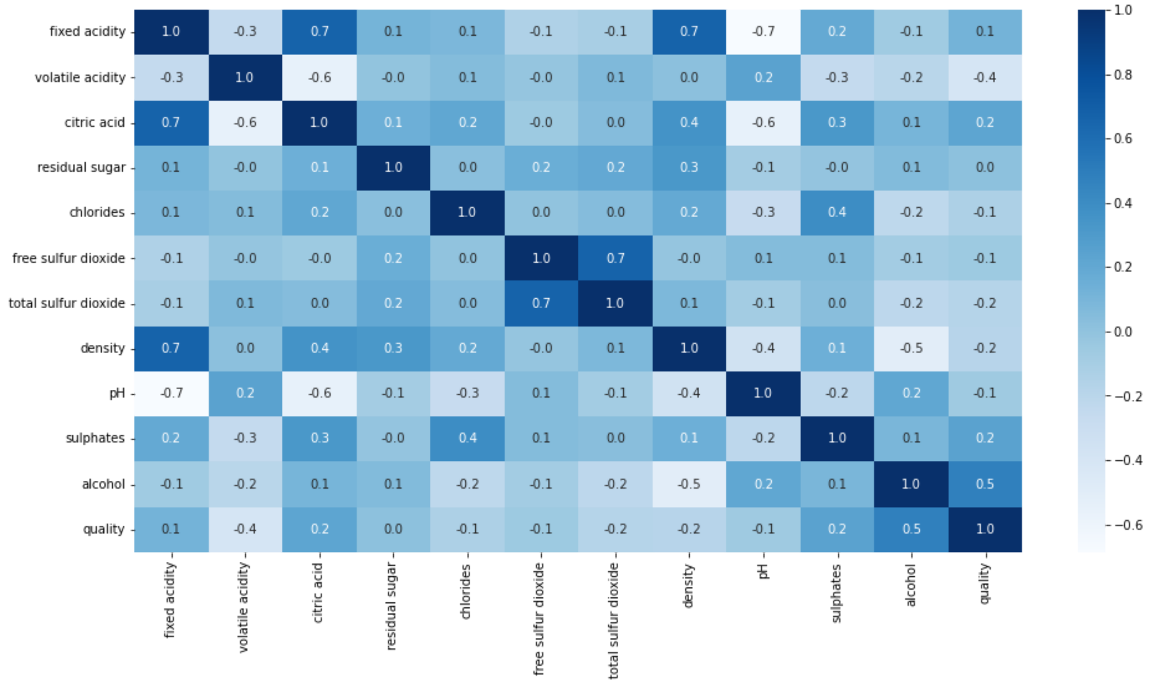
In [27]:
```
print(df["quality"].unique())

df["quality"].value_counts()
```

```
[5 6 7 4 8 3]
```

Out[27]:
```
5    577
6    535
7    167
4     53
8     17
3     10
Name: quality, dtype: int64
```

In [66]:
```python
plt.figure(figsize=(16,8))
sns.heatmap(df.corr(), annot=True, cbar=True, fmt = ".1f", cmap="Blues")
plt.show()
```



In [35]:
```python
X = df.drop(["quality"], axis=1)
y = df["quality"]
```

SMOTE

In [53]:
```python
from imblearn.over_sampling import SMOTE

SS = SMOTE()
x_resampled, y_resampled = SS.fit_resample(X,y)
```

Train test split

In [54]:
```python
x_train, x_test, y_train, y_test = train_test_split(x_resampled, y_resampled
```

In [74]:
```python
import warnings
warnings.filterwarnings("ignore")
```

Logistic Regression

In [73]:
```python
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, y_train)
lr_pred = lr.predict(x_test)
print(accuracy_score(y_test, lr_pred))
```

```
0.49206349206349204
```

Decision Tree Classifier

In [72]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
dt_pred = dt.predict(x_test)
print(accuracy_score(y_test, dt_pred))
```

```
0.7503607503607503
```

Random Forest Classifier

In [71]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train, y_train)
rf_pred = rf.predict(x_test)
print(accuracy_score(y_test, rf_pred))
```

```
0.8196248196248196
```

In [77]:
```
x_test
```

Out[77]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
480	9.400000	0.430000	0.240000	2.800000	0.092000	14.000000	45.000000	0.998000	3.19
2658	7.370167	0.401283	0.272333	2.336167	0.063171	3.680834	10.319166	0.993630	3.45
2835	8.851812	0.358830	0.424819	2.336491	0.100883	6.000000	10.423394	0.995625	3.15
3389	5.884353	0.813810	0.055170	1.516326	0.047585	14.843527	82.442198	0.992629	3.54
2243	6.019599	1.102515	0.113011	1.576704	0.161261	7.000000	19.511358	0.994281	3.46
...	...	...	...	...	...	...	...	...	
1473	7.016976	0.948802	0.000000	2.302960	0.205524	12.022111	23.575606	0.995393	3.43
9	6.700000	0.580000	0.080000	1.800000	0.097000	15.000000	65.000000	0.995900	3.28
1412	7.474099	1.331346	0.000000	3.453434	0.111820	5.000000	12.147520	0.995918	3.58
2096	9.307931	0.636057	0.365375	2.100000	0.077586	12.374451	44.832602	0.998287	3.30
3312	7.968178	0.532046	0.349090	2.460229	0.075716	7.943185	16.943185	0.992479	3.20

693 rows × 11 columns

In [61]:
```
from collections import Counter

print("Before Smoting :", Counter(y))
print("After Smoting :", Counter(y_resampled))
```

```
Before Smoting : Counter({5: 577, 6: 535, 7: 167, 4: 53, 8: 17, 3: 10})
After Smoting : Counter({5: 577, 6: 577, 7: 577, 4: 577, 8: 577, 3: 577})
```

Predictive Modelling

In [68]:

```
import numpy as np
input = (9.400000   0.430000    0.240000    2.800000    0.092000    14.00000

# Changing the input data to numpy array
in_as_np = np.asarray(input)

#Reshaping the numpy array as we are predicting only one instance
in_reshaped = in_as_np.reshape(1,-1)

prediction = rf.predict(in_reshaped)

print(prediction)
```

```
[6]
```