Importing libraries

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import roc_curve, auc
```

Reading and loading the data as pandas read_csv I used for reading the csv file.

In [3]:
```
Data_wine = pd.read_csv('QualityPrediction.csv')
```

Checking the data by using head() command by seeing the first five values with rows and columns of our dataset on which I am going to analyse.

In [4]:
```
Data_wine.head()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quali
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	

shape used for Count the number of rows and columns in our dataset.

In [5]:
```
Data_wine.shape
```

Out[5]:
```
(1599, 12)
```

In [6]:
```
# Getting column names

Data_wine.columns
```

Out[6]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
```

```
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [7]:

```python
# Getting the complete information od data columns.

Data_wine.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 [8]:

```python
# Getting 5 number summary details

Data_wine.describe()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	(
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	(
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	(
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	(
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	(
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	(
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1

In [9]:

```python
# Checking null values

Data_wine.isnull().sum()
```

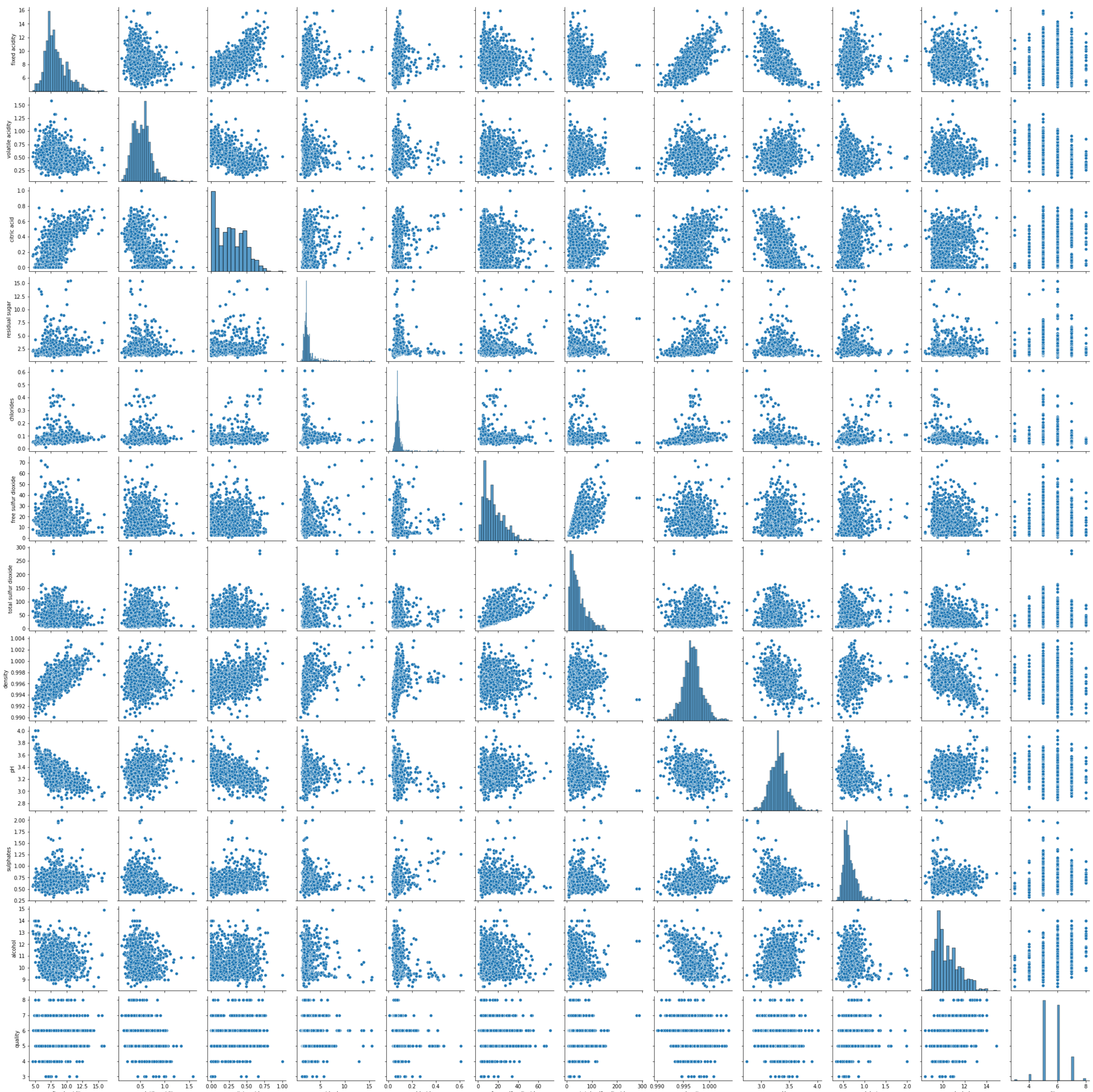
Out[9]:

```
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
```

Data visualization (pairplot)

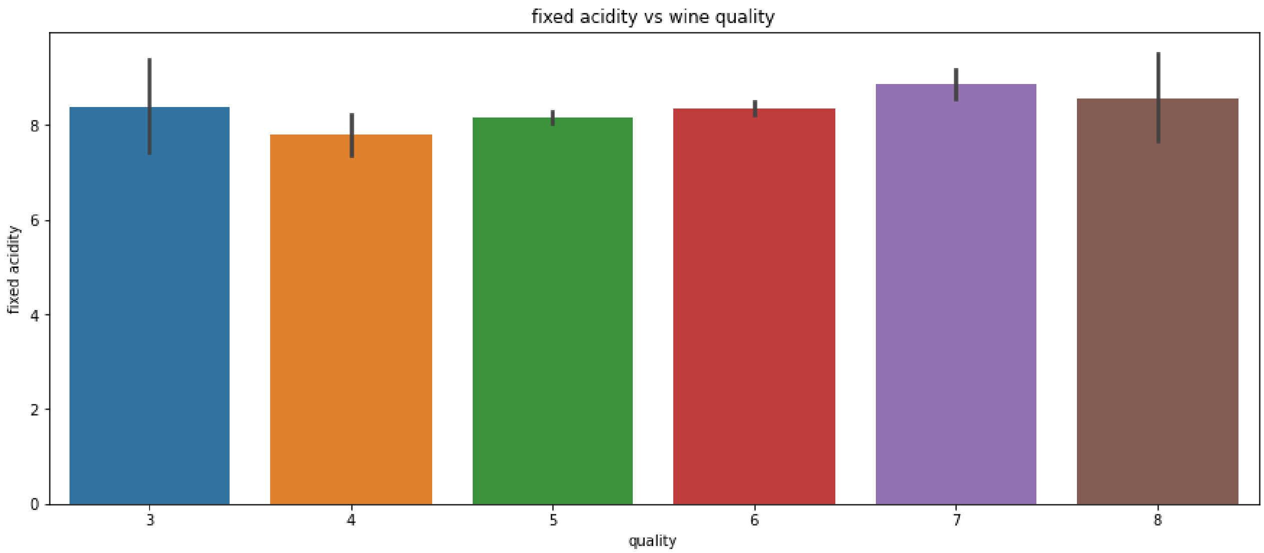
In [10]:
```python
sns.pairplot(Data_wine)
plt.show()
```



Distribution of all the data columns with the target variable using barplot for showing that how the columns are distributed in dataset.

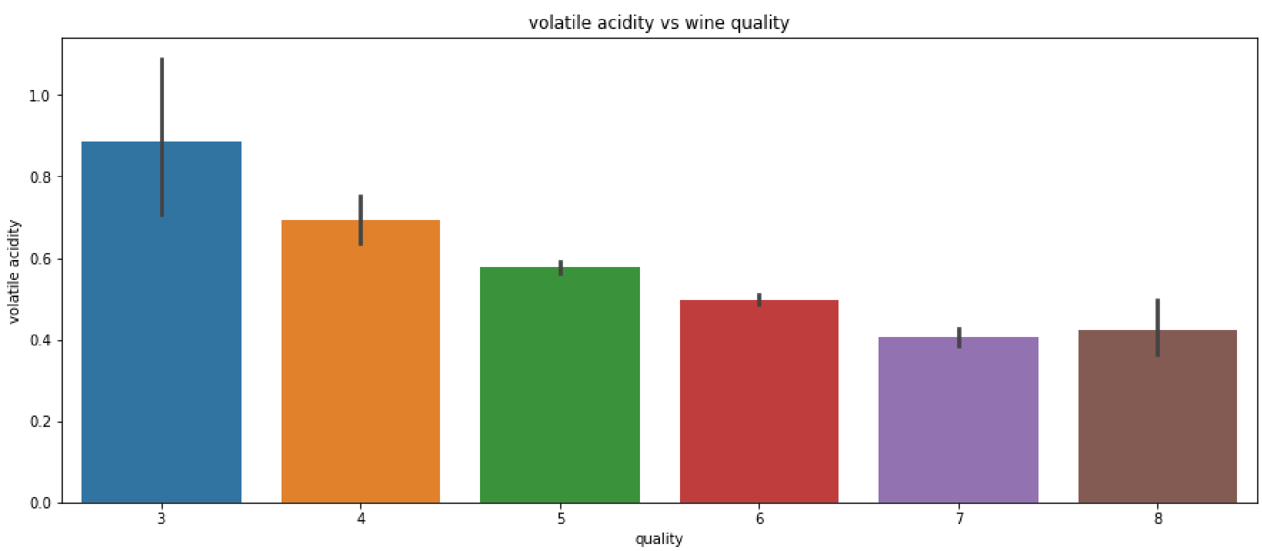
In [11]:
```python
#No proper clarification of fixed acidity with quality of wine.

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'fixed acidity', data = Data_wine)
plt.title('fixed acidity vs wine quality')
```

```
plt.xlabel('quality')
plt.ylabel('fixed acidity')
plt.show()
```



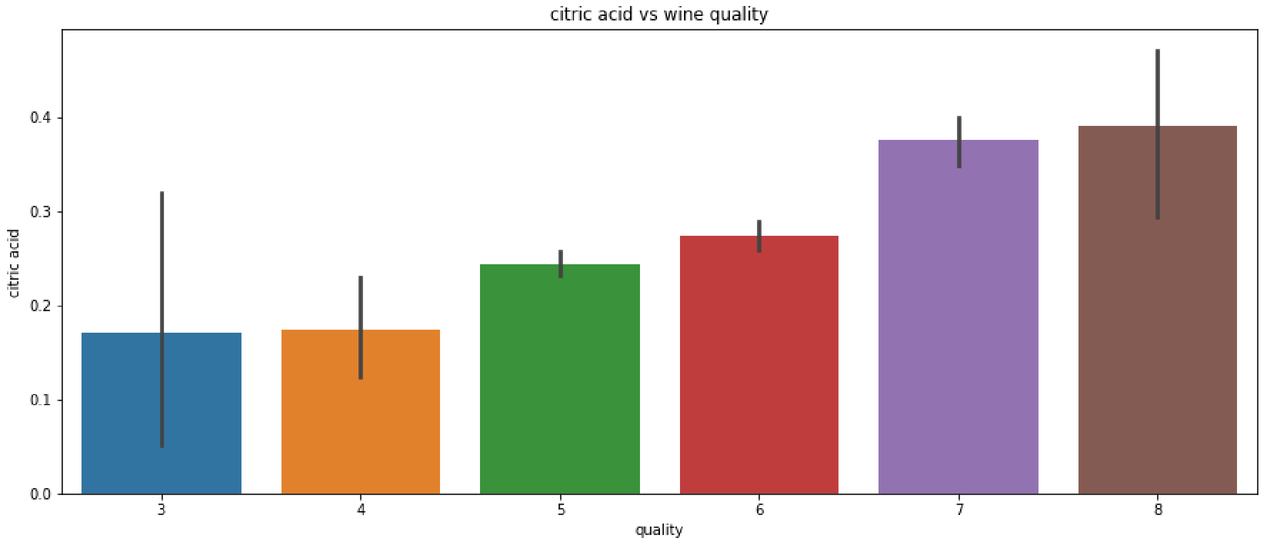
In [12]:

```
# The level of volatile acidity decreasing when the quality of the wine is increasing.


plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'volatile acidity', data = Data_wine)
plt.title('volatile acidity vs wine quality')
plt.xlabel('quality')
plt.ylabel('volatile acidity')
plt.show()
```



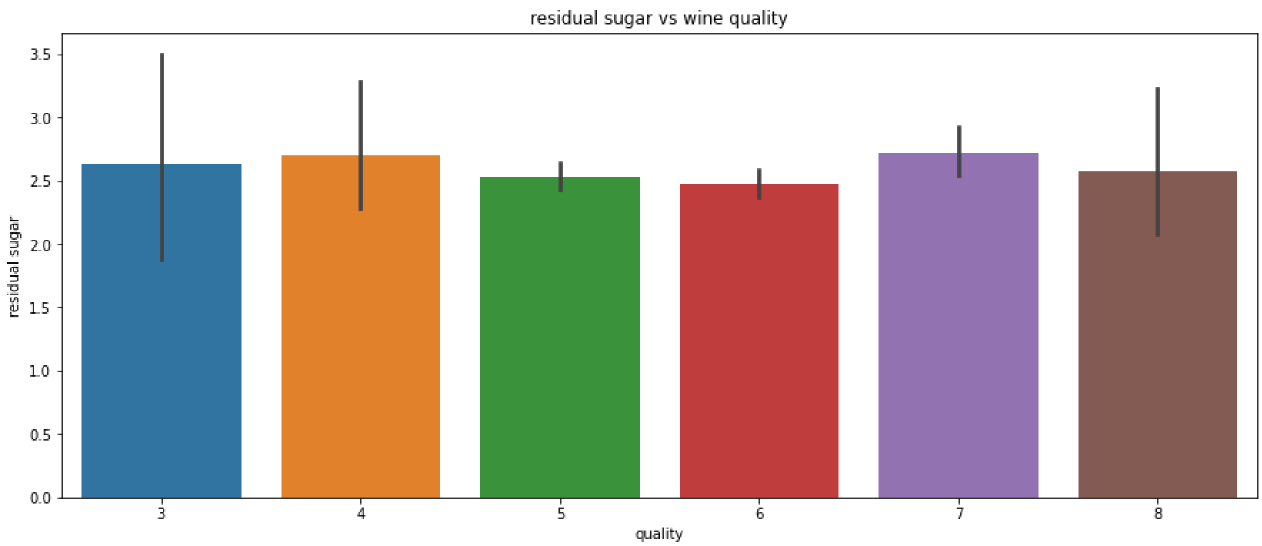
In [13]:

```
# The level of citric acid going higher with the higher quality of wine.

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'citric acid', data = Data_wine)
plt.title('citric acid vs wine quality')
plt.xlabel('quality')
plt.ylabel('citric acid')
plt.show()
```



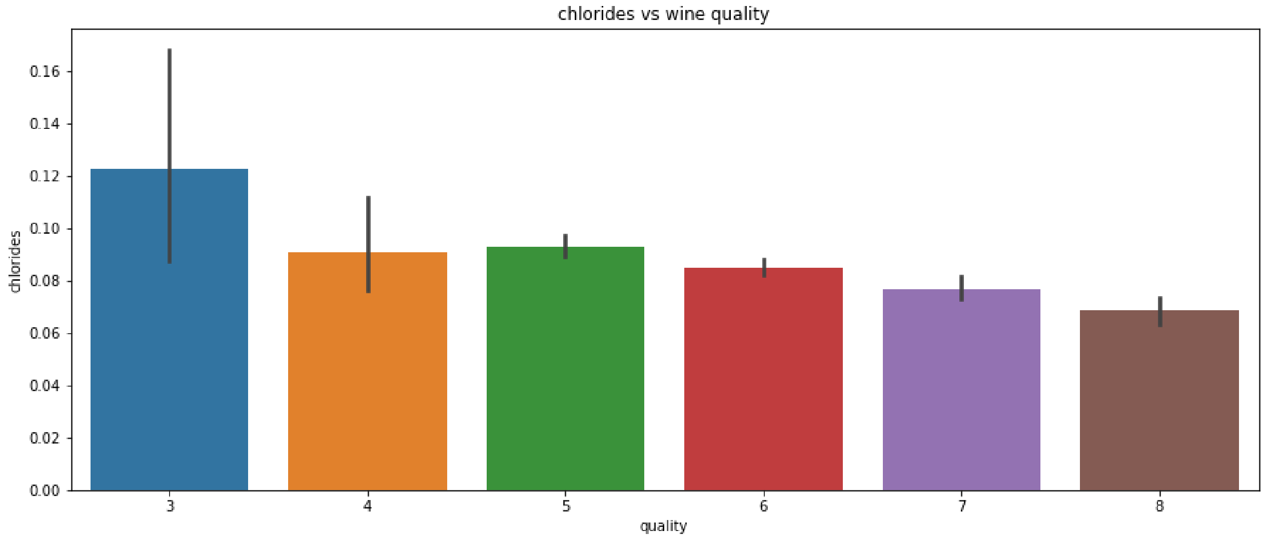
In [14]:

```
# No proper clarification of residual sugar with quality of wine.

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'residual sugar', data = Data_wine)
plt.title('residual sugar vs wine quality')
plt.xlabel('quality')
plt.ylabel('residual sugar')
plt.show()
```

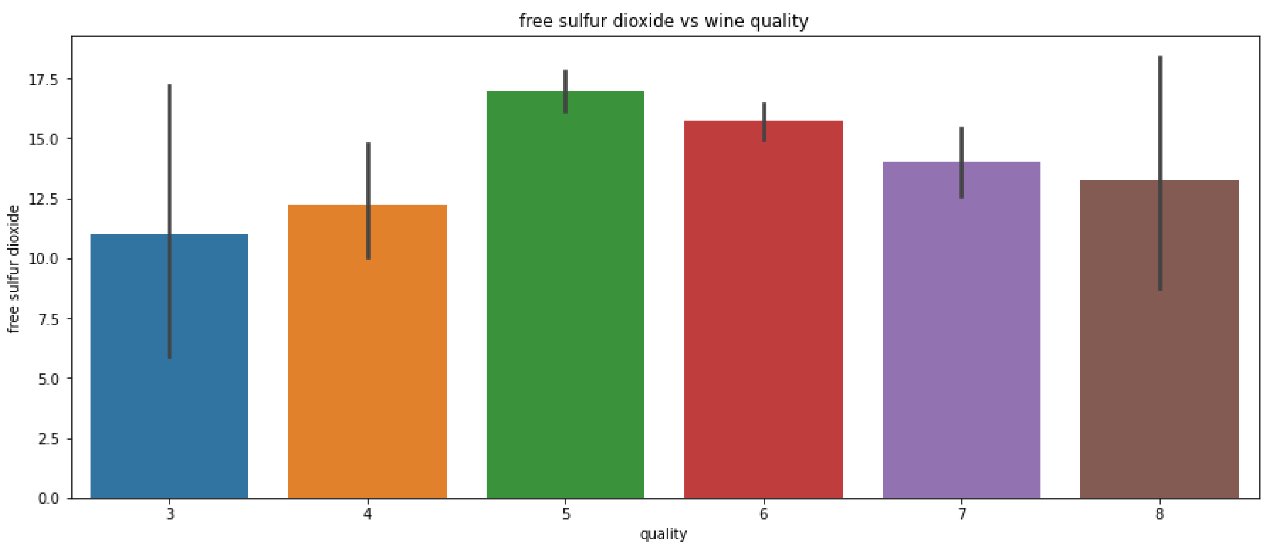


In [15]:

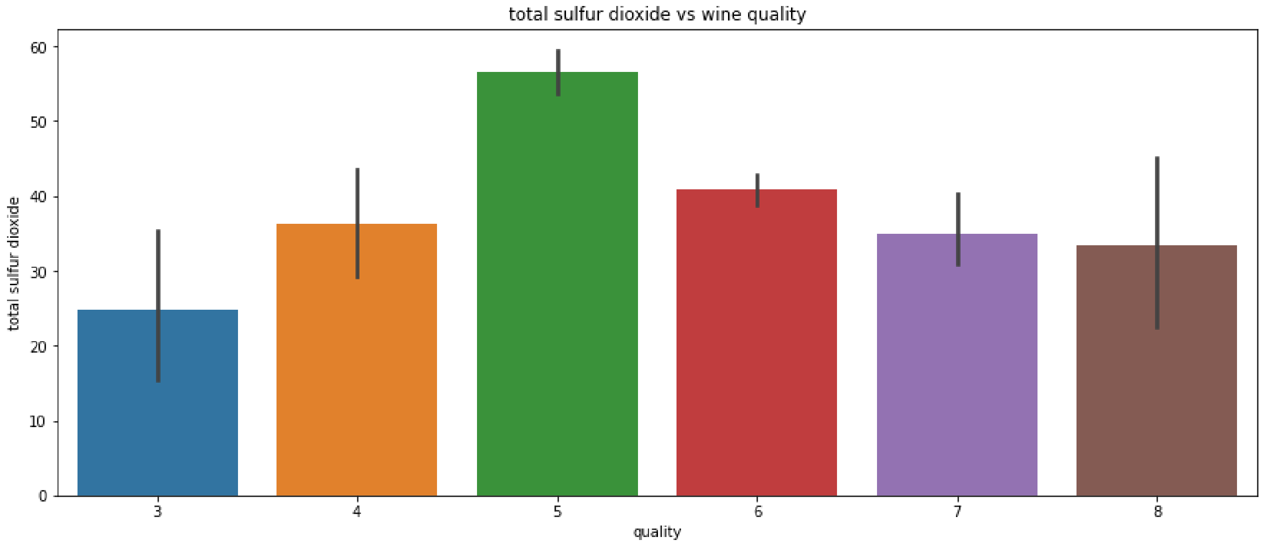
```
# The quantity of chloride going down when the quality of the wine is going higher.

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'chlorides', data = Data_wine)
plt.title('chlorides vs wine quality')
plt.xlabel('quality')
plt.ylabel('chlorides')
plt.show()
```



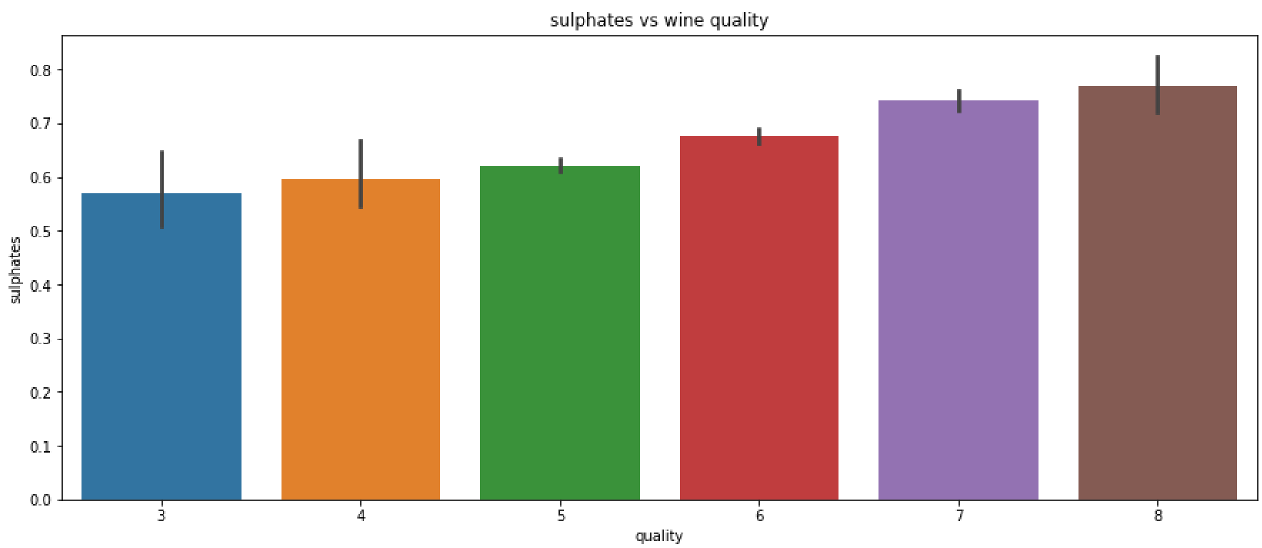
In [16]:
```
# level of free sulfur dioxide with quality

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = Data_wine)
plt.title('free sulfur dioxide vs wine quality')
plt.xlabel('quality')
plt.ylabel('free sulfur dioxide')
plt.show()
```



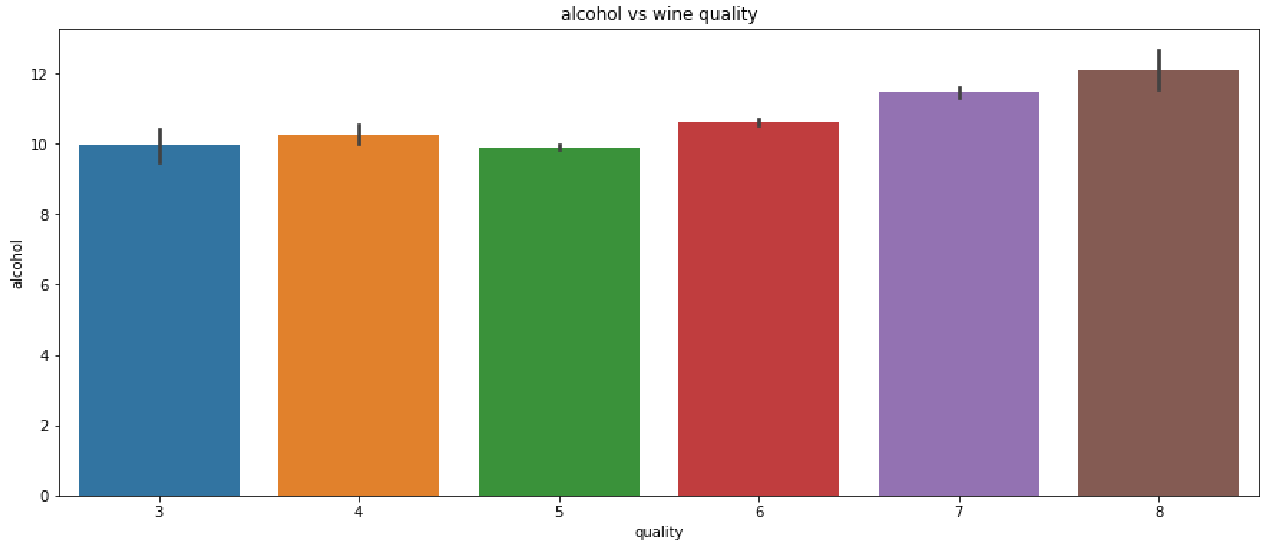
In [17]:
```
# level of total sulfur dioxide with quality

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data = Data_wine)
plt.title('total sulfur dioxide vs wine quality')
plt.xlabel('quality')
plt.ylabel('total sulfur dioxide')
plt.show()
```



In [18]:
```
# level of sulphates increasing with the quality of wine.

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'sulphates', data = Data_wine)
plt.title('sulphates vs wine quality')
plt.xlabel('quality')
plt.ylabel('sulphates')
plt.show()
```

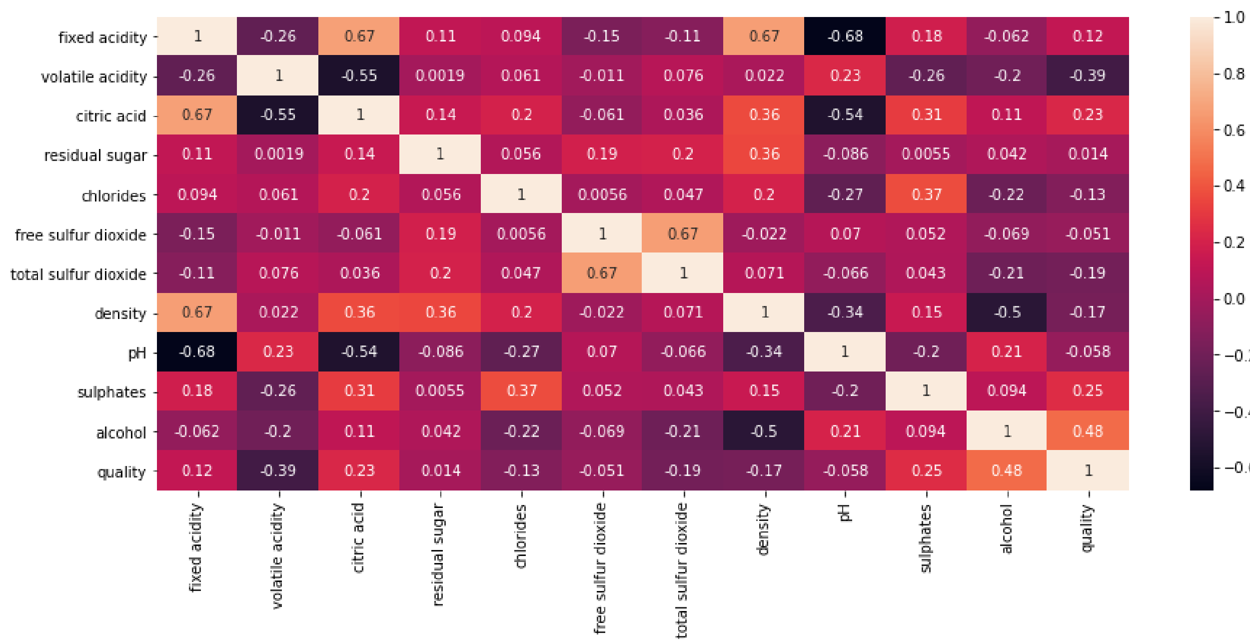


In [19]:
```
# % of alcohol increasing with the quality of wine.

plt.figure(figsize=[15,6])
sns.barplot(x = 'quality', y = 'alcohol', data = Data_wine)
plt.title('alcohol vs wine quality')
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
```



By using the barplot of all the data columns with dependent variable we can see that few independent variables are strongly correlated with dependent variable and few are less correlated with dependent variable so for getting a better understanding I will use correlation matrix so that we can check which independent variable is more correlated with dependent variable

In [20]:
```
plt.figure(figsize=[15,6])
sns.heatmap(Data_wine.corr(),
            annot=True)
plt.show()
```



In [ ]:
```
# We can see by checking the above correlation matrix that alcohal is the best feature
```

preprocessing the data for modeling.

In [21]:
```
Data_wine['quality'].unique()
```

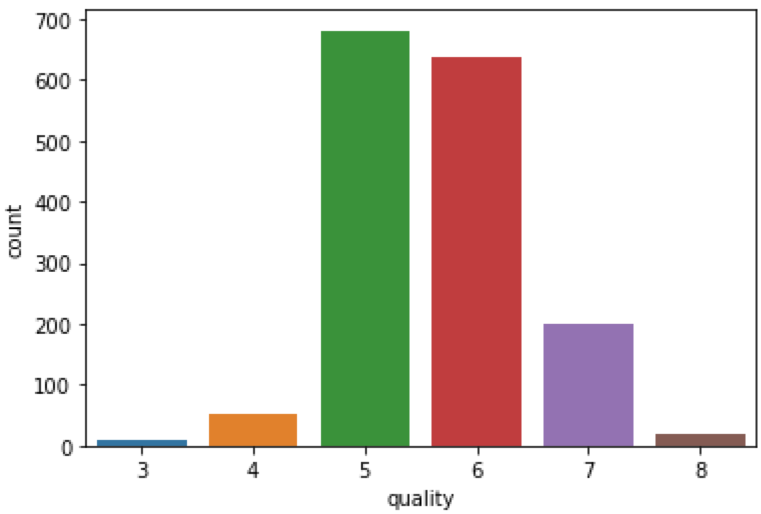
Out[21]:
```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [22]:
```
Data_wine['quality'].value_counts()
```

Out[22]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

In [23]:
```
sns.countplot(x = 'quality' , data = Data_wine)
```

Out[23]: <AxesSubplot:xlabel='quality', ylabel='count'>



In [24]:
```
wine_quality = {3 : 'bad', 4 :'bad', 5: 'bad', 6: 'good', 7: 'good', 8: 'good'}

Data_wine['quality'] = Data_wine['quality'].map(wine_quality)
```

In [25]:
```
Data_wine['quality'].value_counts()
```

Out[25]:
```
good    855
bad     744
Name: quality, dtype: int64
```

In [26]:
```
# Classification into ones and zeroes of the quality of wine.

Data_wine['quality'].replace(['bad','good'] , [0,1] , inplace = True)
```

In [27]:
```
Data_wine['quality'].value_counts()
```

Out[27]:
```
1    855
0    744
Name: quality, dtype: int64
```

In [28]:
```
Data_wine.head()
```

Out[28]:

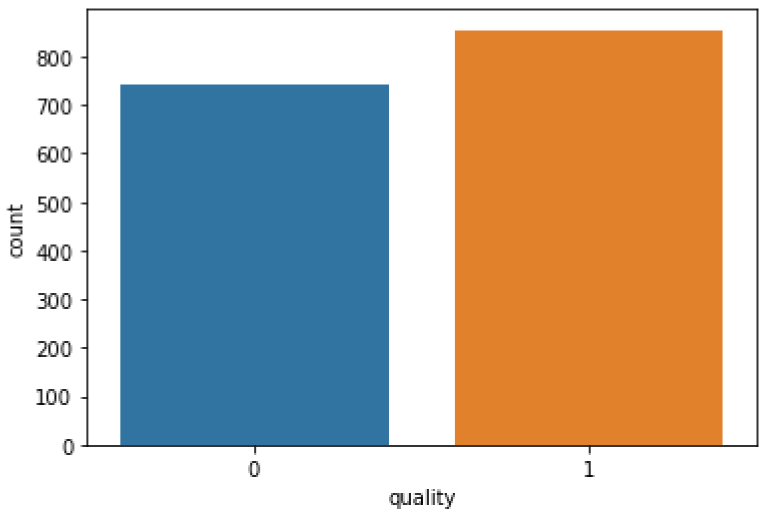
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quali
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quali
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 [29]:
```
sns.countplot(Data_wine['quality'])
```

```
C:\Users\Naveen\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pa
ss the following variable as a keyword arg: x. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

Out[29]: <AxesSubplot:xlabel='quality', ylabel='count'>



Next I am going the seperate the dataset into Independent variables as X and Dependent variable(target variable) as Y

In [30]:
```
X = Data_wine.drop('quality' , axis = 1)
Y = Data_wine['quality']
```

Next I have to split our dataset into train and test data as train will be used to traijn our Model for predicting the wine quality and test data will be used to test or verify the predicting values by the Model

In [31]:
```
# Splitting the data into 80% training data and 20% testing data

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2 , random_state = 0
```

Next , Standardizing the data by using standard scaler as it is the part of preprocessing that transform the data in a way that it will

have mean 0 and standard deviation 1

In [32]:
```
scaler = StandardScaler()

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

Modelling

Next , I will use Random Forest Algorithm to create my Model by using my training data and testing data.

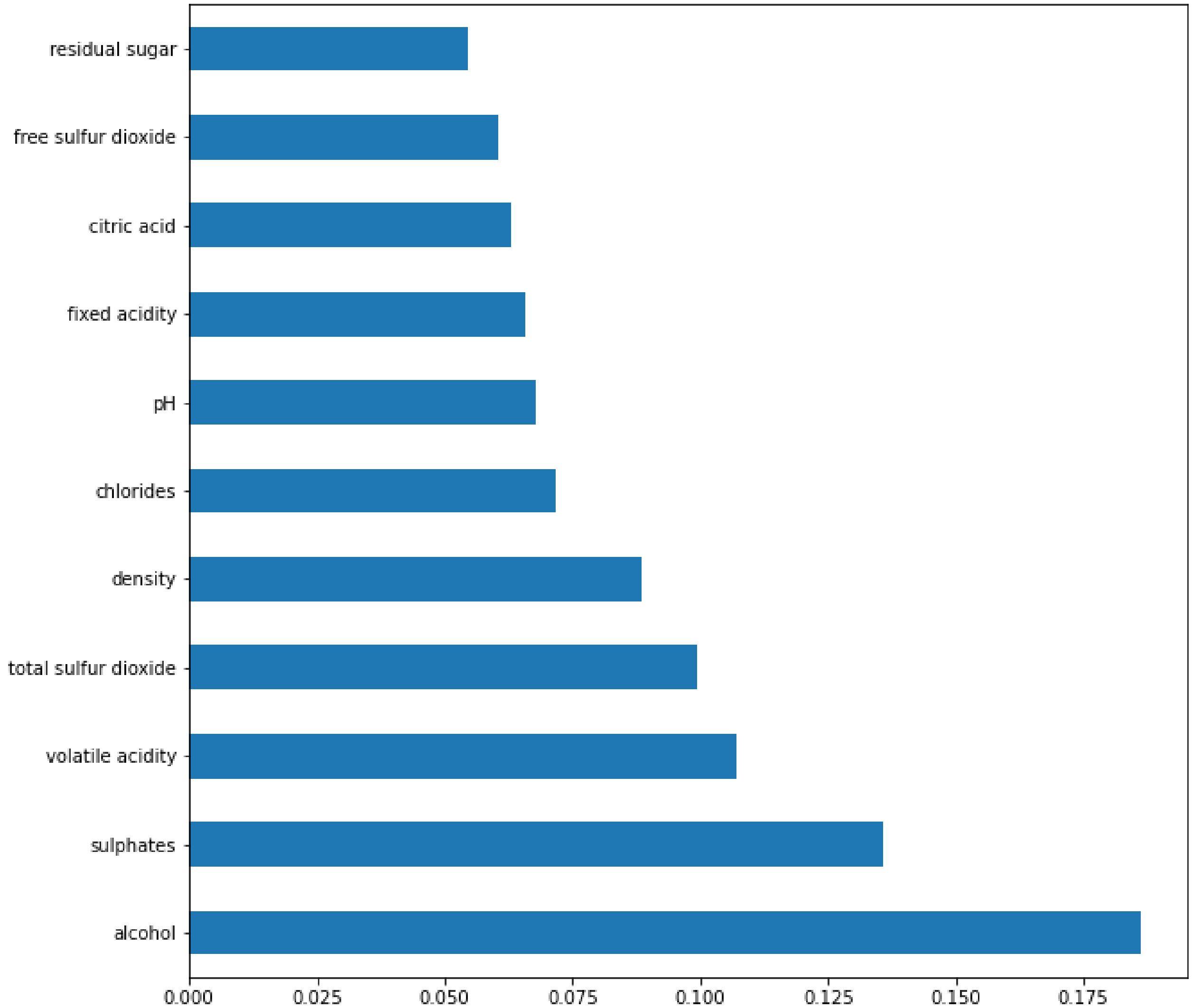
In [33]:
```
Model_rf = RandomForestClassifier(n_estimators = 100 , random_state = 0)
```

In [34]:
```
# Fitting the training set into my model.
Model_rf.fit(X_train , Y_train)

# Accuracy of training data
Model_rf_score = Model_rf.score(X_train , Y_train)

# Training data accuracy
print('Accuracy of training data:' , Model_rf_score)
```

```
Accuracy of training data: 1.0
```

In [35]:
```
variable_importances = pd.Series(Model_rf.feature_importances_, index=X.columns)
variable_importances.nlargest(25).plot(kind='barh',figsize=(10,10))
```

Out[35]:
```
<AxesSubplot:>
```



In [36]:
```
# Predicting the test data
Y_pred = Model_rf.predict(X_test)

# Accuracy of testing data
Model_rf_score = Model_rf.score(X_test , Y_test)

# Testing data accuracy
print('Accuracy of testing data:' , Model_rf_score)
```

```
Accuracy of testing data: 0.821875
```

In [37]:
```
print(Y_pred)
```

```
[0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 1 0 1 1 1 0 0 1 1 0
 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 1
 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1
 0 1 0 1 1 1 1 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 1 1 1
 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 1 1
 0 1 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1
 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0
 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1
 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 1 0 1]
```

In [38]:
```
Y_test = Y_test.values.ravel()
print(Y_test)
```

```
[1 0 1 1 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 1 1 1 0
```

```
 1 1 1 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 1 1 0 1
 0 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 1 1 0 0 0 0 1 0 1
 0 1 0 1 1 1 1 1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 1 1
 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 1 1
 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0
 1 1 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1
 1 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1
 1 0 1 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1]
```

In [39]:

```python
test_error_rate = 1-Model_rf_score
print('Error rate of my model: %0.4f'%test_error_rate)
```

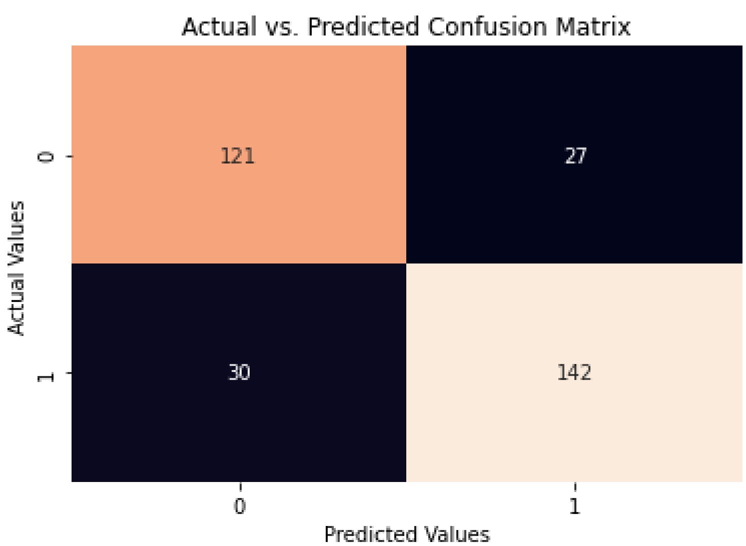
```
Error rate of my model: 0.1781
```

In [40]:

```python
def create_confusion_matrix(Y_test, predictions):
    if (len(Y_test.shape) != len(predictions.shape) == 1):
        return print('Arrays entered are not 1-D.\nPlease enter the correctly sized set
    elif (Y_test.shape != predictions.shape):
        return print('Number of values inside the Arrays are not equal to each other.\n
    else:
        test_crosstb_comp = pd.crosstab(index = Y_test,
                                        columns = predictions)
        test_crosstb = test_crosstb_comp.values
        return test_crosstb
```

In [41]:

```python
confusion_matrix = create_confusion_matrix(Y_test, Y_pred)
sns.heatmap(confusion_matrix, annot=True, fmt='d', cbar=False)
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Actual vs. Predicted Confusion Matrix')
plt.show()
```



In [42]:

```python
# Performance of my model, getting precision , recall , f1 score , support

print(classification_report(Y_test , Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.82      0.81       148
           1       0.84      0.83      0.83       172

    accuracy                           0.82       320
   macro avg       0.82      0.82      0.82       320
```

```
weighted avg       0.82      0.82      0.82       320
```

Predict complete dataset

In [43]:
```
without_target_data_set = Data_wine.iloc[:, Data_wine.columns != 'quality']
print(without_target_data_set)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
...             ...               ...          ...             ...        ...
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol
0         9.4
1         9.8
2         9.8
3         9.8
4         9.4
...       ...
1594     10.5
1595     11.2
1596     11.0
1597     10.2
1598     11.0

[1599 rows x 11 columns]
```

In [44]:
```
with_target_data_set = Data_wine.iloc[:,Data_wine.columns == 'quality']
with_target_data_set = with_target_data_set.values.ravel()
print(with_target_data_set)
```

```
[0 0 0 ... 1 0 1]
```

In [45]:
```
pred_data_sets = Model_rf.predict(without_target_data_set)
```

In [48]:
```
Data_wine['Predict_Data'] = np.where(with_target_data_set,pred_data_sets,with_target_da
```

In [49]:
```
Data_wine['Predict_Data'].replace(to_replace=1, value = 'True' , inplace=True)
Data_wine['Predict_Data'].replace(to_replace=0, value = 'False' ,inplace=True)
```

In [52]:
```
Data_wine.head(15)
```

Out[52]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qua
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	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	
10	6.7	0.580	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	9.2	
11	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	
12	5.6	0.615	0.00	1.6	0.089	16.0	59.0	0.9943	3.58	0.52	9.9	
13	7.8	0.610	0.29	1.6	0.114	9.0	29.0	0.9974	3.26	1.56	9.1	
14	8.9	0.620	0.18	3.8	0.176	52.0	145.0	0.9986	3.16	0.88	9.2	

As per the above output, can say that the original testing values are as much similar to RandomForestClassifier model predicted values as 1 represents the quality greater than 6 which is considered in good quality wine and 0 represents the quality below 6 which is not considered as bad quality wine.

In [53]:
```
Data_wine['quality'].replace(to_replace=1, value = 'Good' , inplace=True)
Data_wine['quality'].replace(to_replace=0, value = 'Bad' , inplace=True)
```

In [54]:
```
Data_wine.head()
```

Out[54]:

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

In [ ]:
```
# Data_wine.to_csv('Predictive_Data.csv')
```

In [ ]:

In [ ]: