```
import numpy as np
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Wine-Quality-Dataset/master/winequality-red.csv",sep = ';')
```

```
df.head()
```

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

```
df['quality'].unique()
```

```
array([5, 6, 7, 4, 8, 3])
```

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

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

```
df.describe()
```

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

```
df.duplicated().sum()
```

```
240
```

```
df=df.drop_duplicates()
```

```
df.duplicated().sum()
```

```
0
```

```
df.columns
```

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

```
x=df.drop("quality",axis=1)
```

```
x
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.
5	7.4	0.660	0.00	1.8	0.075	13.0	40.0	0.99780	3.
...	...	...	...	...	...	...	...	...	
1593	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.

```
y=df['quality']
```

```
y
```

```
0       5
1       5
2       5
3       6
5       5
       ..
1593    6
1594    5
1595    6
1597    5
1598    6
Name: quality, Length: 1359, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1359 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1359 non-null   float64
 1   volatile acidity      1359 non-null   float64
 2   citric acid           1359 non-null   float64
 3   residual sugar        1359 non-null   float64
 4   chlorides             1359 non-null   float64
 5   free sulfur dioxide   1359 non-null   float64
 6   total sulfur dioxide  1359 non-null   float64
 7   density               1359 non-null   float64
 8   pH                    1359 non-null   float64
 9   sulphates             1359 non-null   float64
 10  alcohol               1359 non-null   float64
 11  quality               1359 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 138.0 KB
```

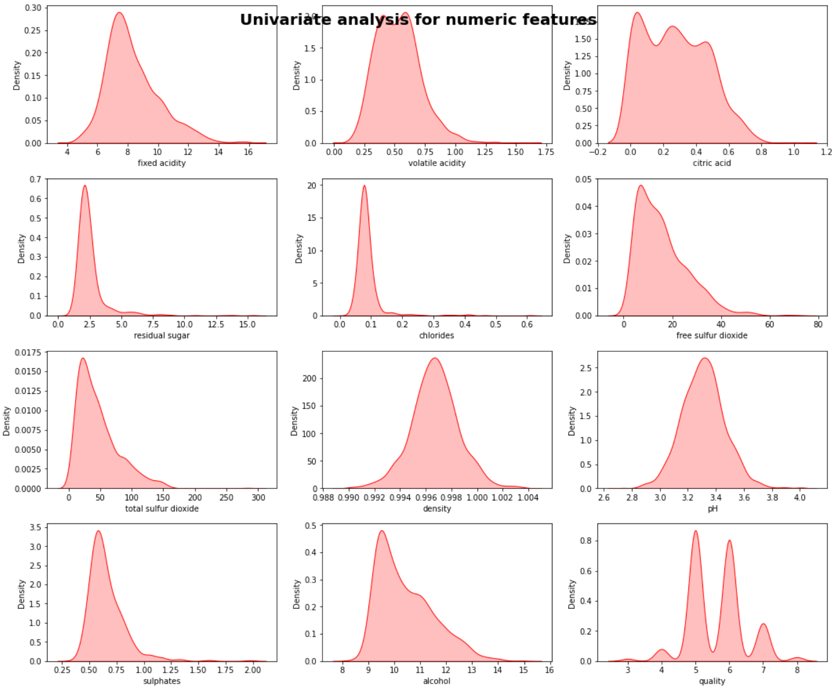
```
df.isnull().sum()
```

```
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
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

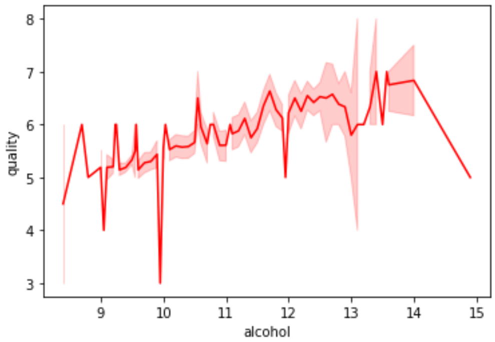
```
plt.figure(figsize=(15,15))
plt.suptitle('Univariate analysis for numeric features',fontsize=20,fontweight='bold')
```

```
for i in range(0,len(df.columns)):
  plt.subplot(5,3,i+1)
  sns.kdeplot(x=df[df.columns[i]],shade=True, color='r')
  plt.xlabel(df.columns[i])
  plt.tight_layout()
```



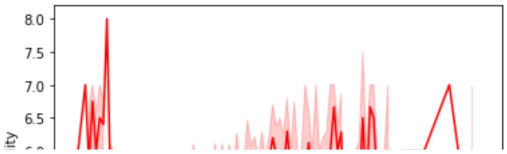
```
sns.lineplot(x='alcohol', y='quality', data=df,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffae1db8430>
```



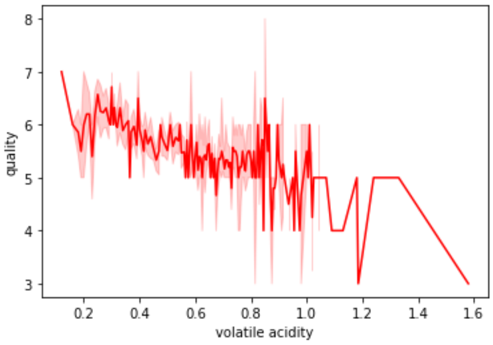
```
sns.lineplot(x='fixed acidity', y='quality', data=df,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffae05198b0>
```



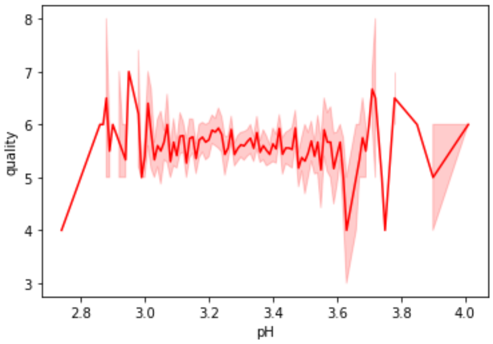
```
sns.lineplot(x='volatile acidity', y='quality', data=df,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffae0502160>
```



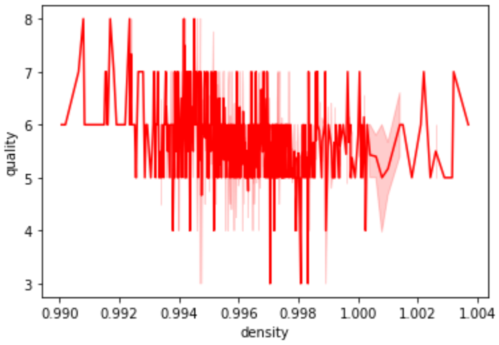
```
sns.lineplot(x='pH', y='quality', data=df,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffae0453ee0>
```



```
sns.lineplot(x='density', y='quality', data=df,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffae03c8fd0>
```

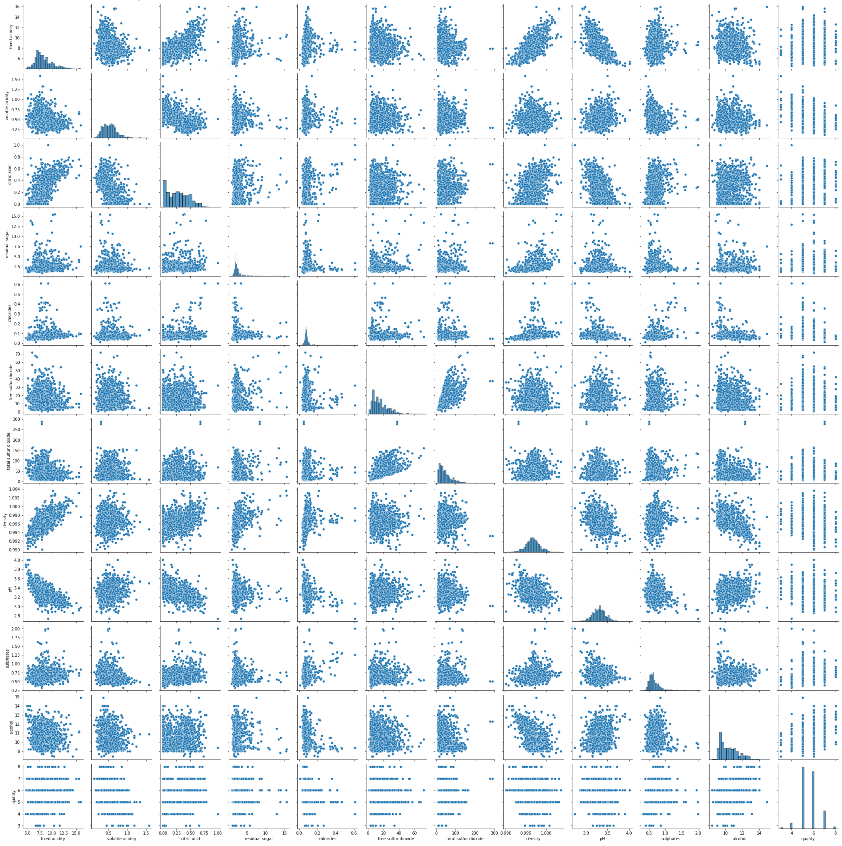


```
df.corr()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
fixed acidity	1.000000	-0.255124	0.667437	0.111025	0.085886	-0.140580	-0.103777
volatile acidity	-0.255124	1.000000	-0.551248	-0.002449	0.055154	-0.020945	0.071701
citric acid	0.667437	-0.551248	1.000000	0.143892	0.210195	-0.048004	0.047358

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7ffae234b040>
```



```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
vif_data=pd.DataFrame()
```

```
vif_data['VIF']=[variance_inflation_factor(df.values,i) for i in range(len(df.columns))]
```

```
vif_data['features']=df.columns
```

```
vif_data
```

	VIF	features
0	75.023033	fixed acidity
1	17.387181	volatile acidity
2	9.195827	citric acid
3	4.915782	residual sugar
4	6.440176	chlorides
5	6.442192	free sulfur dioxide
6	6.601411	total sulfur dioxide
7	1547.276977	density
8	1102.707051	pH
9	22.810607	sulphates
10	146.378710	alcohol
11	74.885884	quality

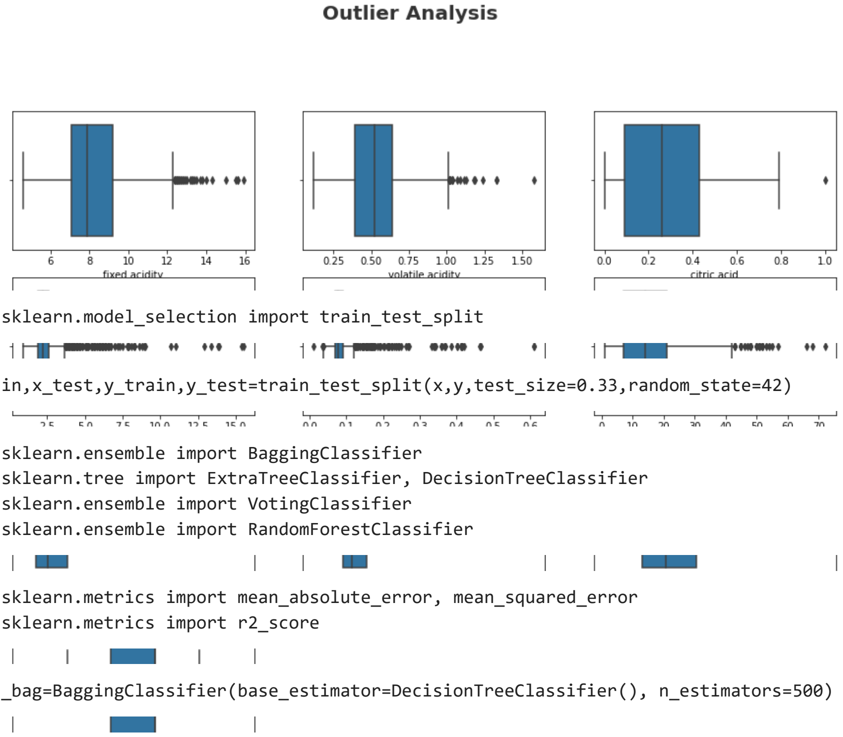
```
df.drop(columns=['density','pH'],axis=1,inplace=True)
```

```
df.columns
```

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

```
plt.figure(figsize=(15,15))
plt.suptitle('Outlier Analysis',fontsize=20, fontweight='bold',alpha=0.8)
for i in range(len(df.columns)):
  plt.subplot(5,3,i+1)
  sns.boxplot(df[df.columns[i]])
  plt.tight_layout
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
```



```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)

from sklearn.ensemble import BaggingClassifier
from sklearn.tree import ExtraTreeClassifier, DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import r2_score

model_bag=BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=500)

extra_tree=ExtraTreeClassifier()

random=RandomForestClassifier()

dtr=DecisionTreeClassifier()

voting_reg=VotingClassifier(estimators=[('r3',extra_tree),('r4',random),('r5',dtr),('r6',model_bag)])

params={'r3__max_depth':[2,3],'r3__min_samples_split':[3],'r3__min_samples_leaf':[2],'r4__n_estimators':[200],'r4__max_depth':[2,3],'r4__
        'r4__min_samples_split':[3],'r5__max_depth':[2,3], 'r5__min_samples_split':[3],'r5__min_samples_leaf':[2]}

from sklearn.model_selection import GridSearchCV

grid=GridSearchCV(estimator=voting_reg,param_grid=params,verbose=3,n_jobs=-1,cv=10)

grid.fit(x_train,y_train)
```

```
Fitting 10 folds for each of 8 candidates, totalling 80 fits
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_split.py:676: UserWarning: The least populated class in y has only
  warnings.warn(
GridSearchCV(cv=10,
             estimator=VotingClassifier(estimators=[('r3',
                                                     ExtraTreeClassifier()),
                                                    ('r4',
                                                     RandomForestClassifier()),
                                                    ('r5',
                                                     DecisionTreeClassifier()),
                                                    ('r6',
                                                     BaggingClassifier(base_estimator=DecisionTreeClassifier(),
                                                                       n_estimators=500))]),
             n_jobs=-1,
             param_grid={'r3__max_depth': [2, 3], 'r3__min_samples_leaf': [2],
                         'r3__min_samples_split': [3], 'r4__max_depth': [2, 3],
                         'r4__min_samples_leaf': [2],
                         'r4__min_samples_split': [3],
                         'r4__n_estimators': [200], 'r5__max_depth': [2, 3],
                         'r5__min_samples_leaf': [2],
                         'r5__min_samples_split': [3]},
             verbose=3)
```

```
grid.best_score_
```

```
0.5725274725274725
```

```
new_svr=grid.best_params_
```

```
new_svr
```

```
{'r3__max_depth': 2,
 'r3__min_samples_leaf': 2,
 'r3__min_samples_split': 3,
 'r4__max_depth': 3,
 'r4__min_samples_leaf': 2,
 'r4__min_samples_split': 3,
 'r4__n_estimators': 200,
 'r5__max_depth': 2,
 'r5__min_samples_leaf': 2,
 'r5__min_samples_split': 3}
```

```
final_model=voting_reg=VotingClassifier(estimators=[('r3',ExtraTreeClassifier(max_depth=2,min_samples_split=3,min_samples_leaf=2)),('r4',
```

```
final_model.fit(x_train,y_train)
```

```
VotingClassifier(estimators=[('r3',
                              ExtraTreeClassifier(max_depth=2,
                                                  min_samples_leaf=2,
                                                  min_samples_split=3)),
                             ('r4',
                              RandomForestClassifier(max_depth=3,
                                                     min_samples_leaf=3,
                                                     min_samples_split=3,
                                                     n_estimators=200)),
                             ('r5',
                              DecisionTreeClassifier(max_depth=3,
                                                     min_samples_leaf=2,
                                                     min_samples_split=3))])
```

```
y_pred=final_model.predict(x_test)
```

```
r2_score(y_test,y_pred)
```

```
0.041408913968435934
```

✓ 0s completed at 7:11 PM