https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
from warnings import filterwarnings
filterwarnings(action='ignore')
```

Loading Dataset

```
wine = pd.read_csv("/content/winequality-red.csv")
wine.sample(25)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
507	11.2	0.670	0.55	2.3	0.084	6.0	13.0	1.00000	3.17	0.71	9.5	6
547	10.6	0.310	0.49	2.5	0.067	6.0	21.0	0.99870	3.26	0.86	10.7	6
911	9.1	0.280	0.46	9.0	0.114	3.0	9.0	0.99901	3.18	0.60	10.9	6
1502	7.3	0.585	0.18	2.4	0.078	15.0	60.0	0.99638	3.31	0.54	9.8	5
6	7.9	0.600	0.06	1.6	0.069	15.0	59.0	0.99640	3.30	0.46	9.4	5
155	7.1	0.430	0.42	5.5	0.071	28.0	128.0	0.99730	3.42	0.71	10.5	5
1414	10.0	0.320	0.59	2.2	0.077	3.0	15.0	0.99940	3.20	0.78	9.6	5
833	11.6	0.470	0.44	1.6	0.147	36.0	51.0	0.99836	3.38	0.86	9.9	4
797	9.3	0.370	0.44	1.6	0.038	21.0	42.0	0.99526	3.24	0.81	10.8	7
1180	8.2	0.350	0.33	2.4	0.076	11.0	47.0	0.99599	3.27	0.81	11.0	6
1162	8.5	0.320	0.42	2.3	0.075	12.0	19.0	0.99434	3.14	0.71	11.8	7
555	15.5	0.645	0.49	4.2	0.095	10.0	23.0	1.00315	2.92	0.74	11.1	5
1562	7.2	0.695	0.13	2.0	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1313	7.0	0.360	0.21	2.3	0.086	20.0	65.0	0.99558	3.40	0.54	10.1	6
562	9.0	0.540	0.49	2.9	0.094	41.0	110.0	0.99820	3.08	0.61	9.2	5
657	12.0	0.500	0.59	1.4	0.073	23.0	42.0	0.99800	2.92	0.68	10.5	7
564	13.0	0.470	0.49	4.3	0.085	6.0	47.0	1.00210	3.30	0.68	12.7	6
130	8.0	0.745	0.56	2.0	0.118	30.0	134.0	0.99680	3.24	0.66	9.4	5
1250	7.1	0.600	0.01	2.3	0.079	24.0	37.0	0.99514	3.40	0.61	10.9	6
1105	6.3	0.570	0.28	2.1	0.048	13.0	49.0	0.99374	3.41	0.60	12.8	5
1155	8.3	0.600	0.25	2.2	0.118	9.0	38.0	0.99616	3.15	0.53	9.8	5
1382	8.0	0.600	0.22	2.1	0.080	25.0	105.0	0.99613	3.30	0.49	9.9	5
1023	8.2	0.320	0.42	2.3	0.098	3.0	9.0	0.99506	3.27	0.55	12.3	6
346	6.6	0.815	0.02	2.7	0.072	17.0	34.0	0.99550	3.58	0.89	12.3	7

```
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
```

Description

```
wine.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
count	1599.000000	1599.000000	1599.000000	1599.000000	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	46.467792	0.996747	3.311113	0.658149
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000

Finding Null Values

```
wine.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
```

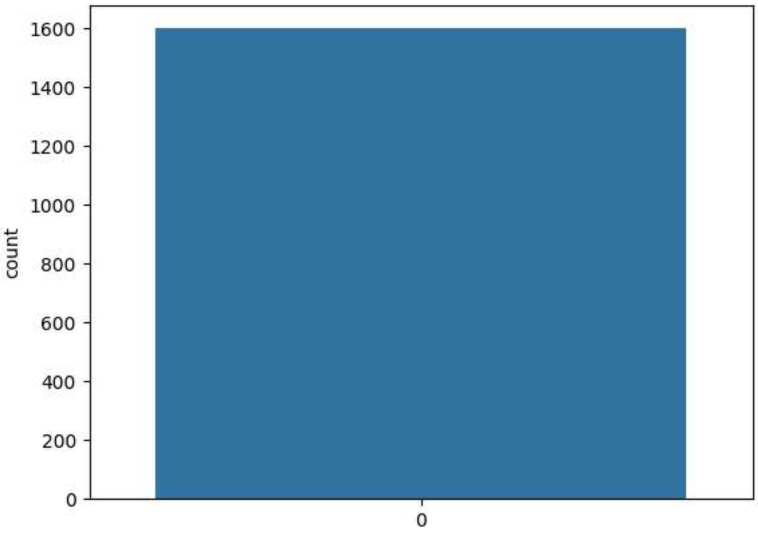
```
wine.groupby('quality').mean()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
quality											
3	8.360000	0.884500	0.171000	2.635000	0.122500	11.000000	24.900000	0.997464	3.398000	0.570000	9.955000
4	7.779245	0.693962	0.174151	2.694340	0.090679	12.264151	36.245283	0.996542	3.381509	0.596415	10.265094
5	8.167254	0.577041	0.243686	2.528855	0.092736	16.983847	56.513950	0.997104	3.304949	0.620969	9.899706
6	8.347179	0.497484	0.273824	2.477194	0.084956	15.711599	40.869906	0.996615	3.318072	0.675329	10.629519
7	8.872362	0.403920	0.375176	2.720603	0.076588	14.045226	35.020101	0.996104	3.290754	0.741256	11.465913

Data Analysis

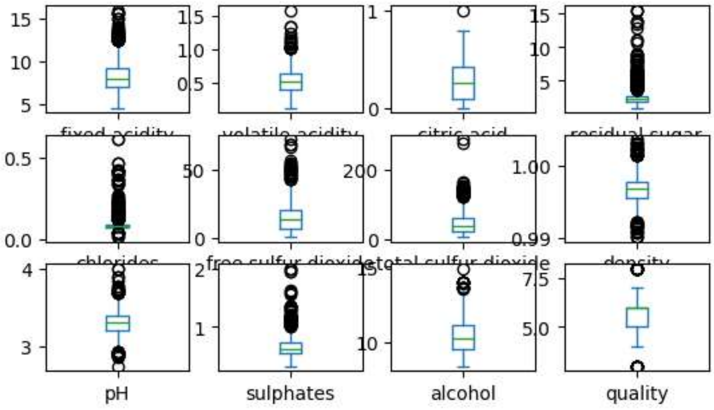
Countplot:

```
sns.countplot(wine['quality'])
plt.show()
```



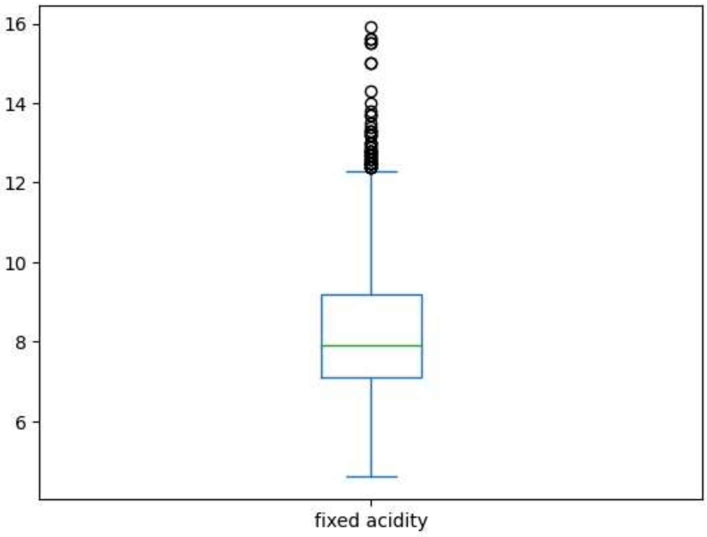
```
wine.plot(kind ='box',subplots = True, layout =(4,4),sharex = False)
```

```
fixed acidity               Axes(0.125,0.712609;0.168478x0.167391)
volatile acidity        Axes(0.327174,0.712609;0.168478x0.167391)
citric acid             Axes(0.529348,0.712609;0.168478x0.167391)
residual sugar          Axes(0.731522,0.712609;0.168478x0.167391)
chlorides                   Axes(0.125,0.511739;0.168478x0.167391)
free sulfur dioxide     Axes(0.327174,0.511739;0.168478x0.167391)
total sulfur dioxide    Axes(0.529348,0.511739;0.168478x0.167391)
density                 Axes(0.731522,0.511739;0.168478x0.167391)
pH                           Axes(0.125,0.31087;0.168478x0.167391)
sulphates                Axes(0.327174,0.31087;0.168478x0.167391)
alcohol                  Axes(0.529348,0.31087;0.168478x0.167391)
quality                  Axes(0.731522,0.31087;0.168478x0.167391)
dtype: object
```



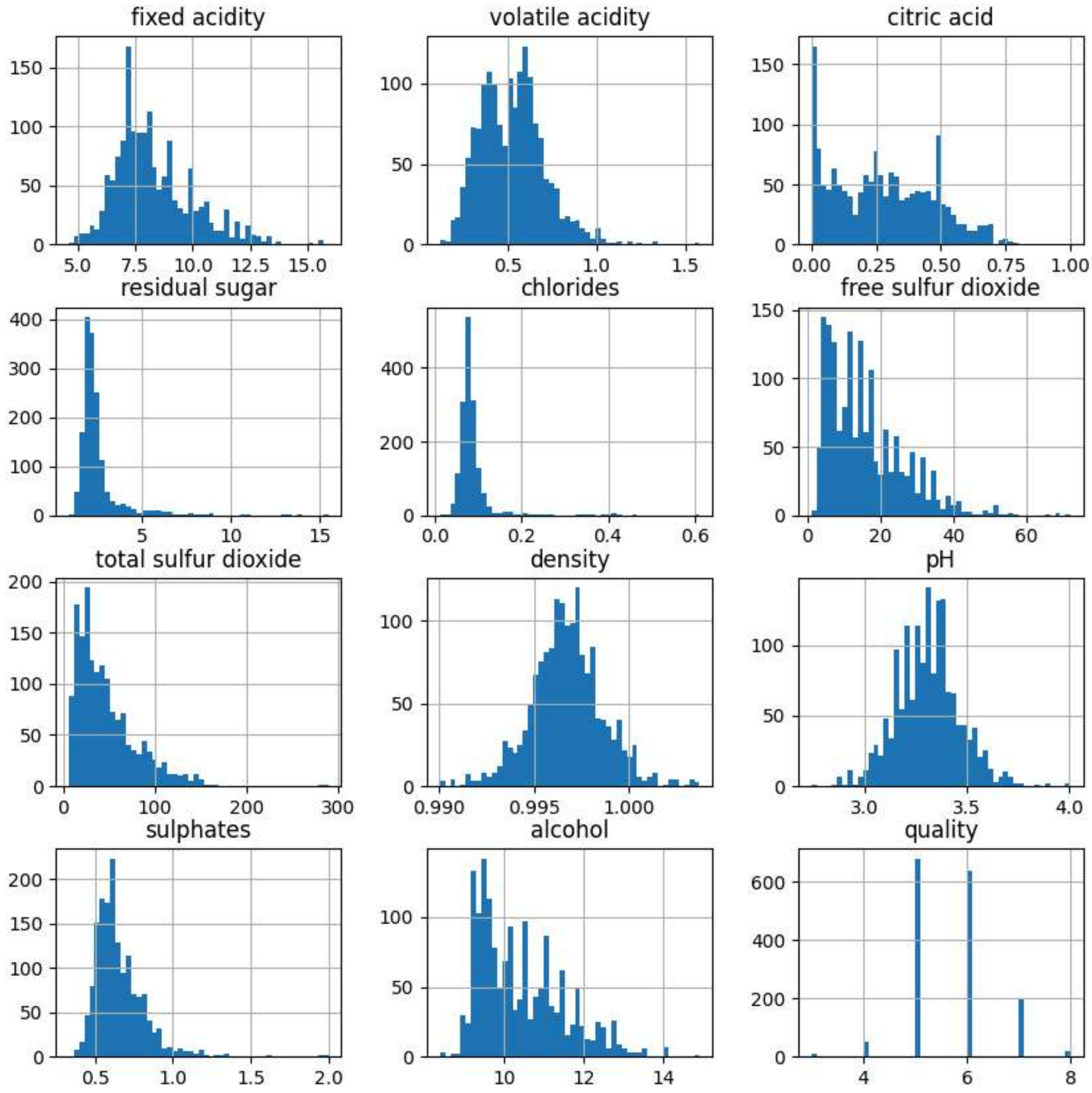
```
wine['fixed acidity'].plot(kind ='box')
```

```
<Axes: >
```



Histogram

```
wine.hist(figsize=(10,10),bins=50)
plt.show()
```



Feature Selection

```
wine.sample(5)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
215	7.0	0.49	0.49	5.6	0.060	26.0	121.0	0.99740	3.34	0.76	10.5	5
84	6.3	0.30	0.48	1.8	0.069	18.0	61.0	0.99590	3.44	0.78	10.3	6
1208	7.2	0.36	0.46	2.1	0.074	24.0	44.0	0.99534	3.40	0.85	11.0	7
1587	5.8	0.61	0.11	1.8	0.066	18.0	28.0	0.99483	3.55	0.66	10.9	6

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

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

```
# If wine quality is 7 or above then will consider as good quality wine
wine['goodquality'] = [1 if x >= 7 else 0 for x in wine['quality']]
wine.sample(5)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	goodquality
539	11.2	0.50	0.74	5.15	0.100	5.0	17.0	0.99960	3.22	0.62	11.2	5	0
630	8.7	0.54	0.26	2.50	0.097	7.0	31.0	0.99760	3.27	0.60	9.3	6	0
922	8.4	0.62	0.12	1.80	0.072	38.0	46.0	0.99504	3.38	0.89	11.8	6	0
1365	7.8	0.50	0.09	2.20	0.115	10.0	42.0	0.99710	3.18	0.62	9.5	5	0

```
# See total number of good vs bad wines samples
wine['goodquality'].value_counts()
```

```
0    1382
1     217
Name: goodquality, dtype: int64
```

```
# Separate depedent and indepedent variables
X = wine.drop(['quality','goodquality'], axis = 1)
Y = wine['goodquality']
```

```
X
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 11 columns

```
print(Y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: goodquality, Length: 1599, dtype: int64
```

Feature Importance

```
from sklearn.ensemble import ExtraTreesClassifier
classifiern = ExtraTreesClassifier()
classifiern.fit(X,Y)
score = classifiern.feature_importances_
print(score)
```

```
[0.07642049 0.10357498 0.09302055 0.07172338 0.06930915 0.07053006
 0.08113963 0.08497152 0.06915126 0.11090966 0.16924932]
```

Splitting Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state=7)
```

Result

```
model_res=pd.DataFrame(columns=['Model', 'Score'])
```

LogisticRegression:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score,confusion_matrix
# accuracy_score(Y_test,Y_pred)
model_res.loc[len(model_res)] = ['LogisticRegression', accuracy_score(Y_test,y_pred)]
model_res
```

```
                Model    Score
0  LogisticRegression  0.86875
```

Using KNN:

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
model_res.loc[len(model_res)] = ['KNeighborsClassifier', accuracy_score(Y_test,y_pred)]
model_res
```

```
                  Model     Score
0    LogisticRegression  0.870833
1  KNeighborsClassifier  0.872917
```

Using SVC:

```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
model_res.loc[len(model_res)] = ['SVC', accuracy_score(Y_test,y_pred)]
model_res
```

```
Accuracy Score: 0.86875
                  Model     Score
0    LogisticRegression  0.870833
1  KNeighborsClassifier  0.872917
2                   SVC  0.868750
```

Using Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
model_res.loc[len(model_res)] = ['DecisionTreeClassifier', accuracy_score(Y_test,y_pred)]
model_res
```

```
Accuracy Score: 0.8645833333333334
                    Model     Score
0      LogisticRegression  0.870833
1    KNeighborsClassifier  0.872917
2                     SVC  0.868750
3  DecisionTreeClassifier  0.864583
```

Using GaussianNB:

```
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
model_res.loc[len(model_res)] = ['GaussianNB', accuracy_score(Y_test,y_pred)]
model_res
```

```
Accuracy Score: 0.8333333333333334
                    Model     Score
0      LogisticRegression  0.870833
1    KNeighborsClassifier  0.872917
2                     SVC  0.868750
3  DecisionTreeClassifier  0.864583
4              GaussianNB  0.833333
```

Using Random Forest:

```
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(X_train, Y_train)
y_pred = model2.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
model_res.loc[len(model_res)] = ['RandomForestClassifier', accuracy_score(Y_test,y_pred)]
model_res
```

```
Accuracy Score: 0.89375
                    Model     Score
0      LogisticRegression  0.870833
1    KNeighborsClassifier  0.872917
2                     SVC  0.868750
3  DecisionTreeClassifier  0.864583
4              GaussianNB  0.833333
5  RandomForestClassifier  0.893750
```

```
# !pip install xgboost
```

Using Xgboost:

```
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(X_train, Y_train)
y_pred = model5.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
model_res.loc[len(model_res)] = ['XGBClassifier', accuracy_score(Y_test,y_pred)]
model_res
```

```
Accuracy Score: 0.8916666666666667
```

	Model	Score
0	LogisticRegression	0.870833
1	KNeighborsClassifier	0.872917
2	SVC	0.868750
3	DecisionTreeClassifier	0.864583
4	GaussianNB	0.833333
5	RandomForestClassifier	0.893750
6	XGBClassifier	0.891667

```
model_res = model_res.sort_values(by='Score', ascending=False)
model_res
```

	Model	Score
5	RandomForestClassifier	0.893750
6	XGBClassifier	0.891667
1	KNeighborsClassifier	0.872917
0	LogisticRegression	0.870833
2	SVC	0.868750
3	DecisionTreeClassifier	0.864583
4	GaussianNB	0.833333

Start coding or generate with AI.