In [112…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [113…

```
df=pd.read_csv("F:\decision tree\winequality-red.csv")
```

In [114…

```
df.head()
```

Out[114…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [115…

```
df.describe()
```

Out[115…

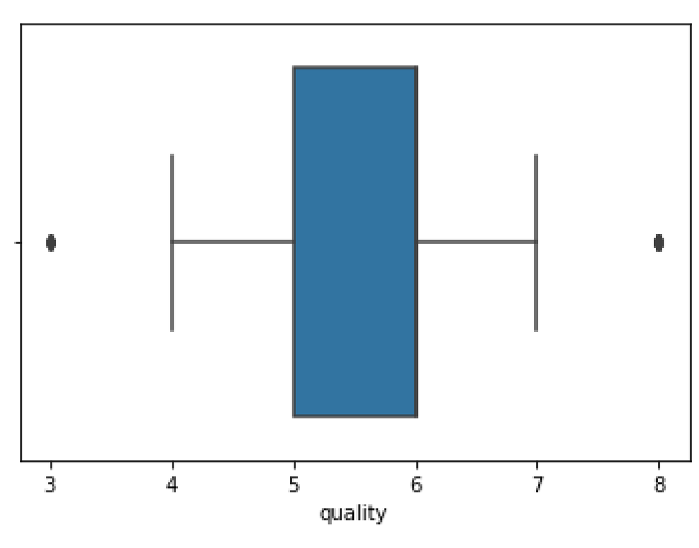
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

In [116…

```
sns.boxplot(x="quality",data=df)
```

Out[116…

```
<AxesSubplot:xlabel='quality'>
```

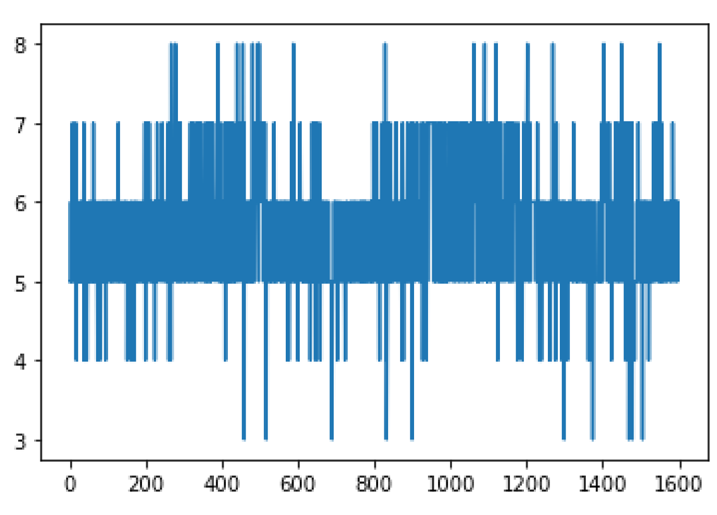


In [117…

```
df.quality.plot()
```

Out[117…

```
<AxesSubplot:>
```



In [118…

```
df.quality.value_counts()
```

Out[118…

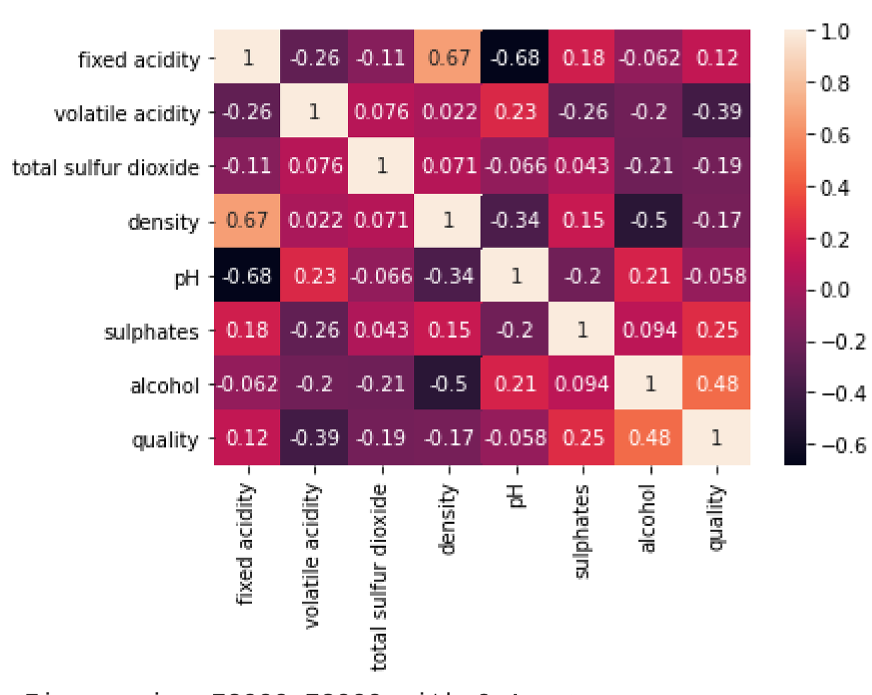
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

In [119…

```
sns.heatmap(df[['fixed acidity','volatile acidity','total sulfur dioxide','density','pH','sulphates','alcohol','quality']].corr(),annot=True)
plt.figure(figsize=(1000,1000))
```

Out[119…

```
<Figure size 72000x72000 with 0 Axes>
```




```
<Figure size 72000x72000 with 0 Axes>
```

In [120…

```
df=df.drop(['volatile acidity','total sulfur dioxide','density','pH','sulphates','alcohol'],axis=1)
plt.figure(figsize=(50,40))
```

Out[120…

```
<Figure size 3600x2880 with 0 Axes>
<Figure size 3600x2880 with 0 Axes>
```

In [121…

```
df.head()
```

Out[121…

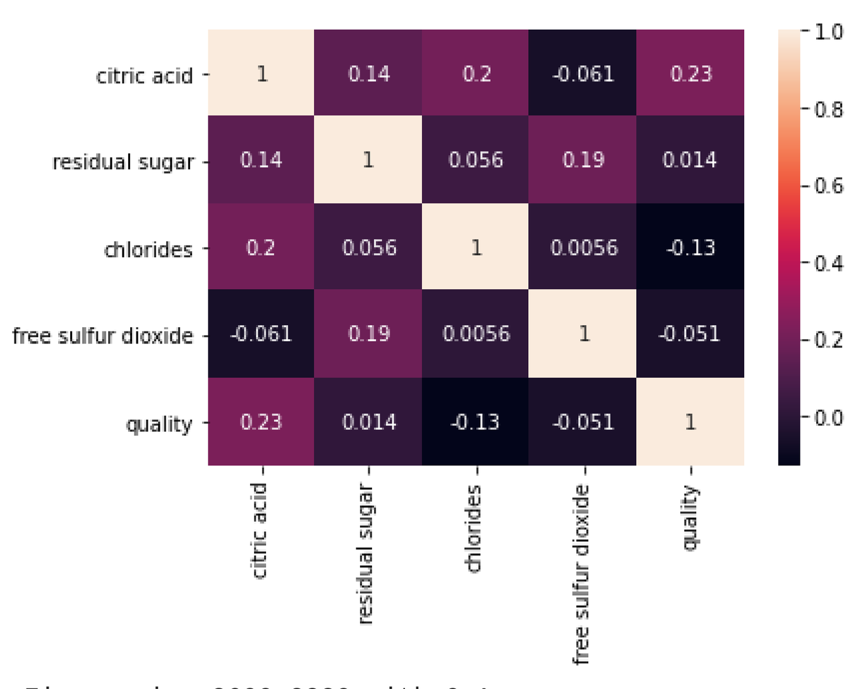
	fixed acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	quality
0	7.4	0.00	1.9	0.076	11.0	5
1	7.8	0.00	2.6	0.098	25.0	5
2	7.8	0.04	2.3	0.092	15.0	5
3	11.2	0.56	1.9	0.075	17.0	6
4	7.4	0.00	1.9	0.076	11.0	5

In [122…

```
sns.heatmap(df[['citric acid','residual sugar','chlorides','free sulfur dioxide','quality']].corr(),annot=True)
plt.figure(figsize=(50,40))
```

Out[122…

```
<Figure size 3600x2880 with 0 Axes>
```




```
<Figure size 3600x2880 with 0 Axes>
```

In [123…

```
df=df.drop(['residual sugar','chlorides','free sulfur dioxide'],axis=1)
df.head()
```

Out[123…

	fixed acidity	citric acid	quality
0	7.4	0.00	5
1	7.8	0.00	5
2	7.8	0.04	5
3	11.2	0.56	6
4	7.4	0.00	5

In [124…

```
y=df['quality'].apply(lambda y:1 if y>=5 else 0)
X=df.drop(['quality'],axis=1)
```

In [125…

```
y
```

Out[125…

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

In [126…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
model = RandomForestClassifier(n_estimators=10000)
model.fit(X_train, y_train)
```

Out[126…

```
RandomForestClassifier(n_estimators=10000)
```

In [127…

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, y_test)
```

In [128…

```
test_data_accuracy
```

Out[128…

```
0.946875
```

In [129…

```
X_test_prediction
```

Out[129…

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [130…

```
X_train
```

Out[130…

	fixed acidity	citric acid
606	9.4	0.48
362	11.9	0.66
533	10.3	0.24
1397	7.3	0.26
733	7.0	0.34
...	...	...
789	8.6	0.17
256	9.4	0.37
968	9.0	0.43
952	8.2	0.40
1273	7.5	0.20

1279 rows × 2 columns

In [138…

```
print(X_test)
```

```
      fixed acidity  citric acid
147             7.6         0.26
937            12.0         0.50
877             7.7         0.01
1000            7.5         0.30
73              8.3         0.26
...             ...          ...
841             6.6         0.00
889            10.7         0.34
1145            8.2         0.43
1476            9.9         0.50
187             7.7         0.10

[320 rows x 2 columns]
```

In [132…

```
model.predict(X_test)
```

Out[132…

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [133…

```
input_data = (7.5,0.5)
_array = np.asarray(input_data)
```

In [135…

```
reshaped = _array.reshape(1,-1)
```

reshaped.head()

In [137…

```
reshaped
```

Out[137…

```
array([[7.5, 0.5]])
```

In [143…

```
C:\Users\Admin Is User\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitte
d with feature names
  warnings.warn(
```

Out[143…

```
array([1], dtype=int64)
```

In [ ]: