Importing the necessary packages

```
import warnings


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.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,accuracy_score


#reading the dataset
df=pd.read_csv("winequality-red.csv")
#displaying the first n lines of the dataset
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	s
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	

```
#finding number of rows and columns
df.shape
```

```
(1599, 12)
```

```
#description of dataset
df.dtypes
```

```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
```

```
quality                 int64
dtype: object
```

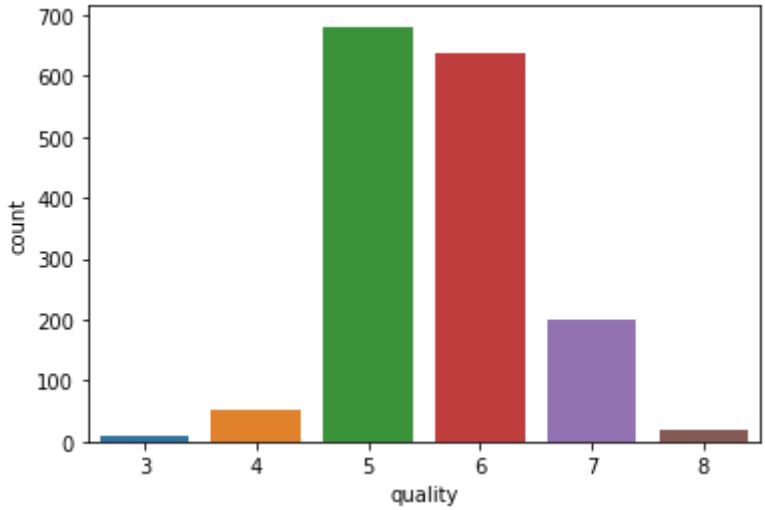
```
#knowing how many null values in data
df.isnull().sum()
```

```
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
```

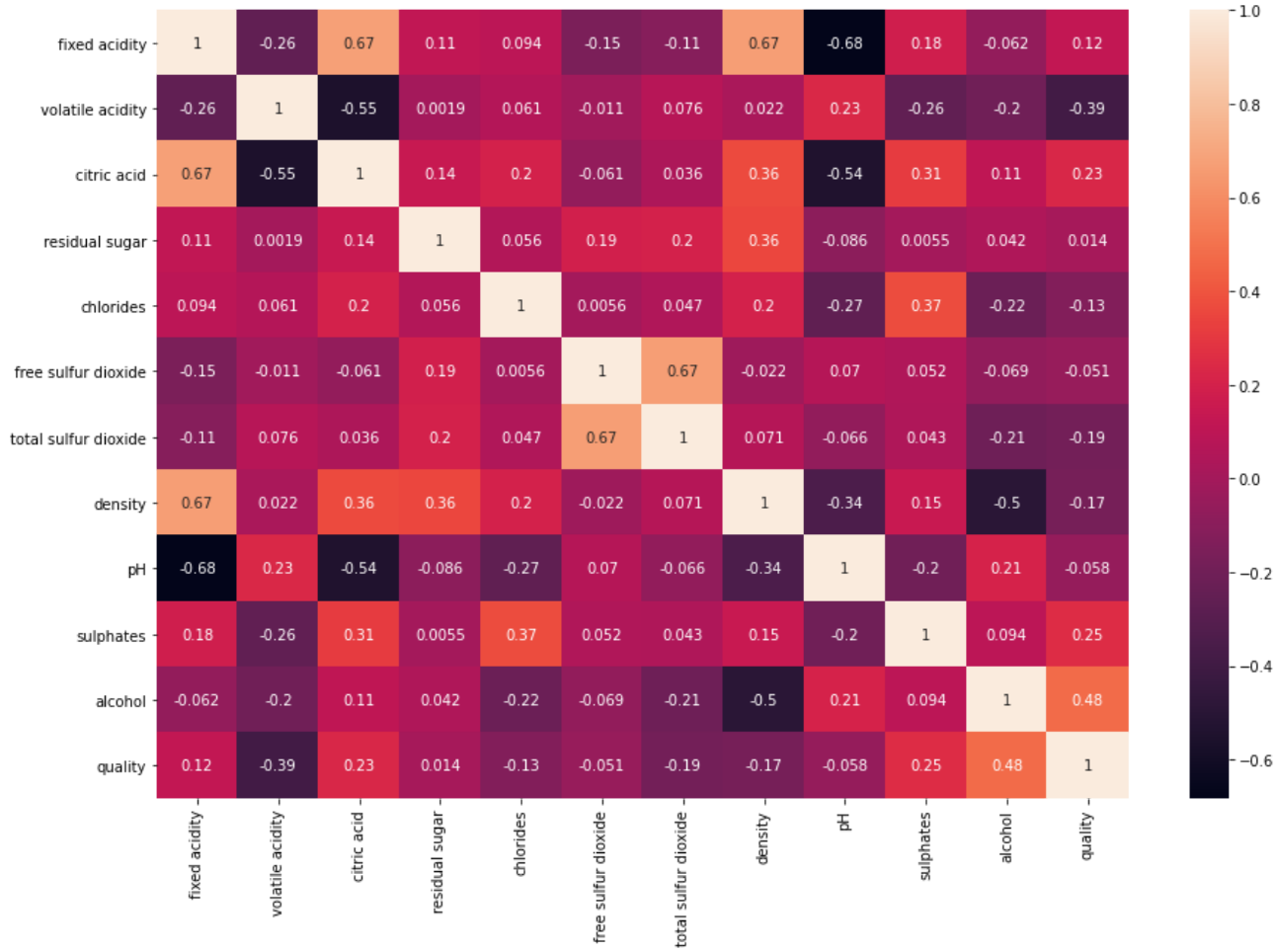
Visualizing quality distribution over the dataset

```
#plotting the histogram with label quality
sns.countplot(df["quality"], data = df)
#displaying the figure (histogram)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
```




```
#drawing multiple plots in one figure of 15x10 inches size
plt.subplots(figsize=(15,10))
#ploting the co-relation heatmap , here annot = true means writing data value in each cell
sns.heatmap(df.corr(), annot = True)
#displaying the heatmap
plt.show()
```



Modifying the quality parameter into low medium and high

```
# the 1st variable stores the values of the quality attribute
lst = df["quality"].values
#creating a list
quality_mod = []
#this list will save the values as low , high , medium
for v in lst:
 if v < 5:
  quality_mod.append("LOW")
 elif  v > 6:
  quality_mod.append("High")
 else:
  quality_mod.append("medium")
```

Label Encoding the modified quality column with values low , medium , high

```
#creating a dataframe with the above list created
quality_mod = pd.DataFrame(data=quality_mod, columns=["category"])
#concatinating the list and dataframe together
data = pd.concat([df, quality_mod], axis=1)
#removing rows and columns, here removing quality column, and overwriting the
#data frame
data.drop(columns="quality", axis=1, inplace=True)
#selecting values excluding last column for parameter X and y
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
#converting the labels into text
le =LabelEncoder()
#labels in column y will be converted into numbers and stored in y
y=le.fit_transform(y)
```

Splitting the dataset into test and train set

```
#spliting the data, without random splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,random_state=
0)
```

First Random forest model is applied

```
#creating random forest classification , with number of trees 250
rf = RandomForestClassifier(n_estimators=250)
#fitting the training the data
rf.fit(X_train, y_train)
#training the model
y_pred_rf = rf.predict(X_test)
```

```
#finding the accuracy
print("Accuracy Random Forest:", rf.score(X_test, y_test))
```

```
Accuracy Random Forest: 0.885
```

Then KNN is applied

```
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred_knn=knn.predict(X_test)
```

```
print("KNN Accuracy:", knn.score(X_test, y_test))
```

```
KNN Accuracy: 0.82
```

Now Desicion Tree

```
model_dt = DecisionTreeClassifier(random_state=1)
model_dt.fit(X_train, y_train)
pred_dt = model_dt.predict(X_test)
print(classification_report(y_test, pred_dt))
#its accuracy increases to 90% after making test data = 0.25
```

```
                   precision    recall  f1-score   support

                0       0.37      0.56      0.45        45
                1       0.25      0.12      0.17        16
                2       0.90      0.86      0.88       339

         accuracy                           0.80       400
        macro avg       0.51      0.51      0.50       400
     weighted avg       0.82      0.80      0.80       400
```