```
#importing all the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#inputing the dataset from UCI machine learning laboratory
headers=['fixed acidity','volatile acidity','citric sugar','residual sugars','chlorides','free
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/wineq
```

```
df.head()
```

	fixed acidity	volatile acidity	citric sugar	residual sugars	chlorides	free sulphur dioxide	total sulphur dioxide	density	pH	sul
0	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
1	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	
2	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	
3	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	
4	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	

```
#datatypes of the attributes
df.shape
```

```
(1600, 12)
```

```
#information about the attributes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600 entries, 0 to 1599
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1600 non-null   object
 1   volatile acidity      1600 non-null   object
 2   citric sugar          1600 non-null   object
 3   residual sugars       1600 non-null   object
 4   chlorides             1600 non-null   object
 5   free sulphur dioxide  1600 non-null   object
 6   total sulphur dioxide 1600 non-null   object
 7   density               1600 non-null   object
 8   pH                    1600 non-null   object
 9   sulphates             1600 non-null   object
 10  alcohol               1600 non-null   object
 11  quality               1600 non-null   object
dtypes: object(12)
memory usage: 150.1+ KB
```

```
# Checking for null values
df.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric sugar            0
residual sugars         0
chlorides               0
free sulphur dioxide    0
total sulphur dioxide   0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
#value counts in the each class
df['quality'].value_counts()
```

```
5          681
6          638
7          199
4           53
```

```
8           18
3           10
quality      1
Name: quality, dtype: int64
```

```
df.describe()
```

	fixed acidity	volatile acidity	citric sugar	residual sugars	chlorides	free sulphur dioxide	total sulphur dioxide	density	pH	sulp
count	1600	1600	1600	1600	1600	1600	1600	1600	1600	
unique	97	144	81	92	154	61	145	437	90	
top	7.2	0.6	0	2	0.08	6	28	0.9972	3.3	
freq	67	47	132	156	66	138	43	36	57	

```
#different unique classes
u=df['quality'].unique()
u
```

```
array(['quality', '5', '6', '7', '4', '8', '3'], dtype=object)
```

```
type(u)
```

```
numpy.ndarray
```

```
len(u)
```

```
7
```

```
df['quality']
```

```
0       quality
1             5
2             5
3             5
4             6
          ...
1595          5
1596          6
1597          6
1598          5
1599          6
Name: quality, Length: 1600, dtype: object
```

```
a = list(range(0, 26))
d = dict(zip(u, a))
d
```

```
{'3': 6, '4': 4, '5': 1, '6': 2, '7': 3, '8': 5, 'quality': 0}
```

```
#converting into values to integer
df['quality'] = df['quality'].map(d)
```

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

```
1    681
2    638
3    199
4     53
5     18
6     10
0      1
Name: quality, dtype: int64
```

```
df['fixed acidity'] = pd.to_numeric(df['fixed acidity'],errors = 'coerce')
df['volatile acidity'] = pd.to_numeric(df['volatile acidity'],errors = 'coerce')
df['citric sugar'] = pd.to_numeric(df['citric sugar'],errors = 'coerce')
df['residual sugars'] = pd.to_numeric(df['residual sugars'],errors = 'coerce')
df['chlorides'] = pd.to_numeric(df['chlorides'],errors = 'coerce')
df['free sulphur dioxide'] = pd.to_numeric(df['free sulphur dioxide'],errors = 'coerce')
df['total sulphur dioxide'] = pd.to_numeric(df['total sulphur dioxide'],errors = 'coerce')
df['density'] = pd.to_numeric(df['density'],errors = 'coerce')
df['pH'] = pd.to_numeric(df['pH'],errors = 'coerce')
df['sulphates'] = pd.to_numeric(df['sulphates'],errors = 'coerce')
```

```
df['pH'] = pd.to_numeric(df['pH'],errors = 'coerce')
df['alcohol'] = pd.to_numeric(df['alcohol'],errors = 'coerce')
df['quality'] = pd.to_numeric(df['quality'],errors = 'coerce')

df = df.dropna()


# create training and testing vars
train = df.sample(frac=0.7, random_state=500)
test = df.drop(train.index)
X_train = train.drop(labels='quality', axis=1)
y_train = train["quality"]
X_test = test.drop(labels='quality', axis=1)
y_test = test['quality']
```

SVM

```
from sklearn import svm


from sklearn.model_selection import cross_val_score,cross_val_predict
clf = svm.SVC(kernel='linear', C=1, random_state=42)
clf.fit(X_train,y_train)
scores = cross_val_score(clf, X_test,y_test, cv=10,scoring='accuracy')
scores
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([0.45833333, 0.58333333, 0.60416667, 0.47916667, 0.58333333,
       0.66666667, 0.6875    , 0.58333333, 0.54166667, 0.5625    ])
```

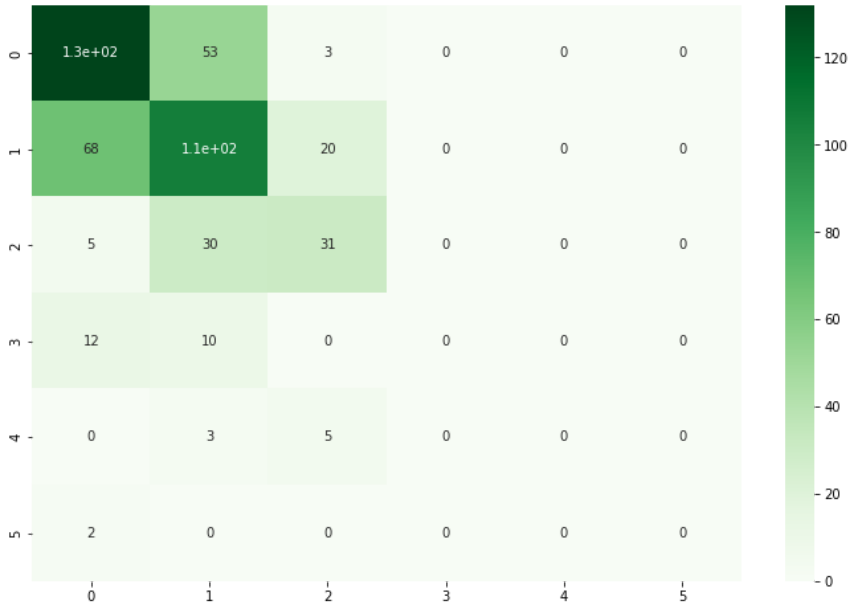
```
from sklearn.metrics import confusion_matrix


#predicting by cross valiadation
predict=cross_val_predict(clf,X_test,y_test,cv=3)
confu=confusion_matrix(y_test,predict)
confu
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([[132,  53,   3,   0,   0,   0],
       [ 68, 106,  20,   0,   0,   0],
       [  5,  30,  31,   0,   0,   0],
       [ 12,  10,   0,   0,   0,   0],
       [  0,   3,   5,   0,   0,   0],
       [  2,   0,   0,   0,   0,   0]])
```

```
plt.figure(figsize=(12,8))
sns.heatmap(confu, cmap="Greens",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7eff61dabcd0>
```



DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier


dtc = DecisionTreeClassifier(random_state=0)
dtc.fit(X_train,y_train)
dtc.score(X_test,y_test)
```

```
0.6145833333333334
```

```
scores = cross_val_score(dtc,X_test,y_test,cv=10, scoring='accuracy')
scores
```

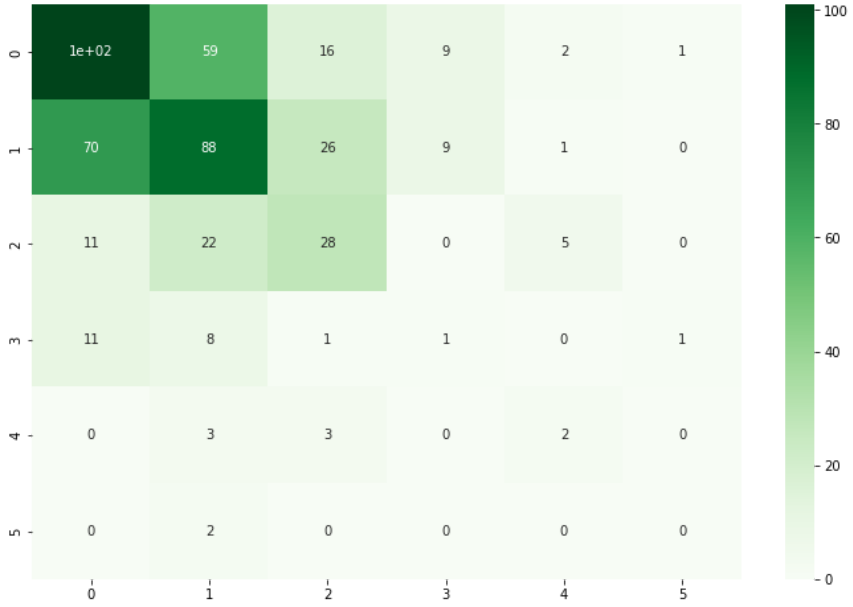
```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([0.375     , 0.4375    , 0.52083333, 0.47916667, 0.4375    ,
       0.47916667, 0.54166667, 0.54166667, 0.41666667, 0.54166667])
```

```
#predicting by cross valiadation
predict=cross_val_predict(dtc,X_test,y_test,cv=3)
confu=confusion_matrix(y_test,predict)
confu
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([[101,  59,  16,   9,   2,   1],
       [ 70,  88,  26,   9,   1,   0],
       [ 11,  22,  28,   0,   5,   0],
       [ 11,   8,   1,   1,   0,   1],
       [  0,   3,   3,   0,   2,   0],
       [  0,   2,   0,   0,   0,   0]])
```

```
plt.figure(figsize=(12,8))
sns.heatmap(confu, cmap="Greens",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7eff56f2cbd0>
```



KNeighborsClassifier

```
#importing the KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier


#training, testing and finding the accuracy
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

```
0.49375
```

```
#cross validation

#finding the cross validation accuarcy
```

```
scores = cross_val_score(knn,X_test,y_test,cv=10, scoring='accuracy')
scores
```

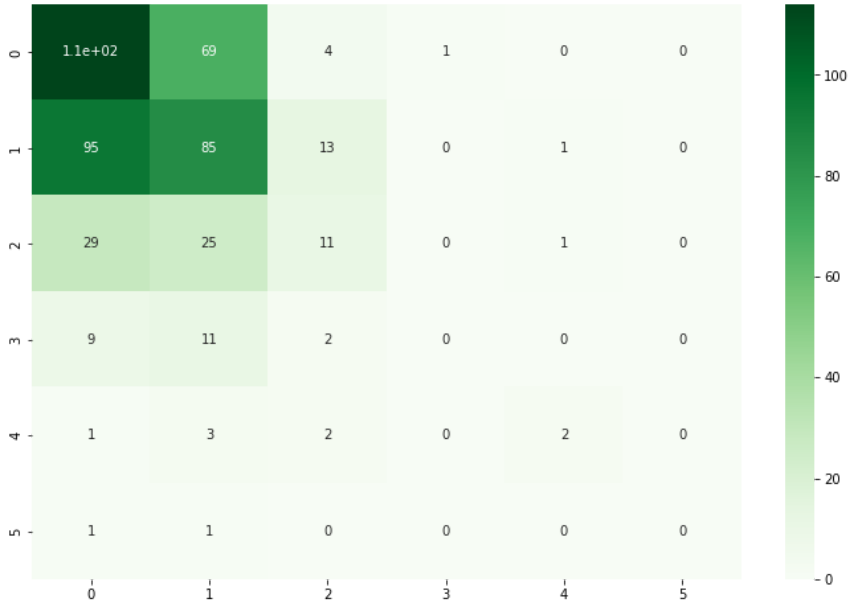
```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([0.375     , 0.39583333, 0.45833333, 0.4375    , 0.45833333,
       0.375     , 0.47916667, 0.27083333, 0.35416667, 0.5       ])
```

```
#predicting by cross valiadation
predict=cross_val_predict(knn,X_test,y_test,cv=3)
confu=confusion_matrix(y_test,predict)
confu
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([[114,  69,   4,   1,   0,   0],
       [ 95,  85,  13,   0,   1,   0],
       [ 29,  25,  11,   0,   1,   0],
       [  9,  11,   2,   0,   0,   0],
       [  1,   3,   2,   0,   2,   0],
       [  1,   1,   0,   0,   0,   0]])
```

```
plt.figure(figsize=(12,8))
sns.heatmap(confu, cmap="Greens",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7eff61dabf50>
```



Random forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
RFC= RandomForestClassifier(max_depth=2, random_state=0)
RFC.fit(X_train,y_train)
RFC.score(X_test,y_test)
```

```
0.5229166666666667
```

```
#cross validation

#finding the cross validation accuarcy
scores = cross_val_score(RFC,X_test,y_test,cv=10, scoring='accuracy')
scores
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([0.39583333, 0.5625    , 0.58333333, 0.39583333, 0.5       ,
       0.60416667, 0.58333333, 0.4375    , 0.58333333, 0.54166667])
```

```
#predicting by cross valiadation
predict=cross_val_predict(RFC,X_test,y_test,cv=3)
confu=confusion_matrix(y_test,predict)
confu
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 2
  UserWarning,
array([[127,  61,   0,   0,   0,   0],
       [ 76, 118,   0,   0,   0,   0],
       [  6,  57,   3,   0,   0,   0],
       [ 13,   9,   0,   0,   0,   0],
       [  0,   8,   0,   0,   0,   0],
       [  1,   1,   0,   0,   0,   0]])
```

```
plt.figure(figsize=(12,8))
sns.heatmap(confu, cmap="Greens",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7eff5560af90>
```