AI AND ML ASSIGNMENT 4

21BCE7255- UPPALAPATI BALAJI

```
import pandas as pd
import numpy as np

df= pd.read_csv("/content/winequality-red.csv")
df.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

```
df.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
```

```
df.shape
```

```
(1599, 12)
```

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

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

```
df.isnull().any()
```

```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
dtype: bool
```

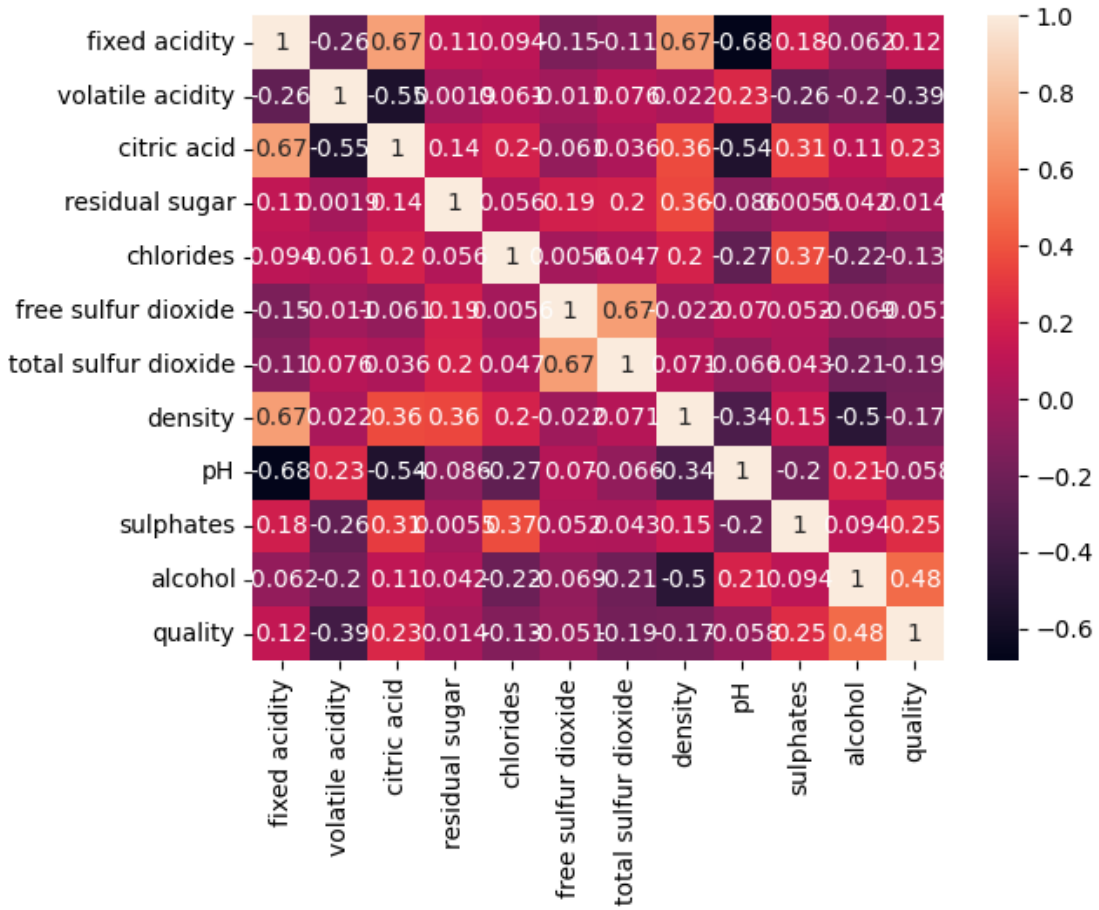
```
df.corr()
```

```
                      fixed acidity  volatile acidity  citric acid  \
fixed acidity              1.000000         -0.256131     0.671703
volatile acidity          -0.256131          1.000000    -0.552496
citric acid                0.671703         -0.552496     1.000000
residual sugar             0.114777          0.001918     0.143577
chlorides                  0.093705          0.061298     0.203823
free sulfur dioxide       -0.153794         -0.010504    -0.060978
total sulfur dioxide      -0.113181          0.076470     0.035533
density                    0.668047          0.022026     0.364947
pH                        -0.682978          0.234937    -0.541904
sulphates                  0.183006         -0.260987     0.312770
alcohol                   -0.061668         -0.202288     0.109903
```

```
quality                    0.124052         -0.390558     0.226373

                      residual sugar  chlorides  free sulfur dioxide  \
fixed acidity               0.114777   0.093705            -0.153794
volatile acidity            0.001918   0.061298            -0.010504
citric acid                 0.143577   0.203823            -0.060978
residual sugar              1.000000   0.055610             0.187049
chlorides                   0.055610   1.000000             0.005562
free sulfur dioxide         0.187049   0.005562             1.000000
total sulfur dioxide        0.203028   0.047400             0.667666
density                     0.355283   0.200632            -0.021946
pH                         -0.085652  -0.265026             0.070377
sulphates                   0.005527   0.371260             0.051658
alcohol                     0.042075  -0.221141            -0.069408
quality                     0.013732  -0.128907            -0.050656

                      total sulfur dioxide   density        pH  sulphates  \
fixed acidity                   -0.113181  0.668047 -0.682978   0.183006
volatile acidity                 0.076470  0.022026  0.234937  -0.260987
citric acid                      0.035533  0.364947 -0.541904   0.312770
residual sugar                   0.203028  0.355283 -0.085652   0.005527
chlorides                        0.047400  0.200632 -0.265026   0.371260
free sulfur dioxide              0.667666 -0.021946  0.070377   0.051658
total sulfur dioxide             1.000000  0.071269 -0.066495   0.042947
density                          0.071269  1.000000 -0.341699   0.148506
pH                              -0.066495 -0.341699  1.000000  -0.196648
sulphates                        0.042947  0.148506 -0.196648   1.000000
alcohol                         -0.205654 -0.496180  0.205633   0.093595
quality                         -0.185100 -0.174919 -0.057731   0.251397

                       alcohol   quality
fixed acidity        -0.061668  0.124052
volatile acidity     -0.202288 -0.390558
citric acid           0.109903  0.226373
residual sugar        0.042075  0.013732
chlorides            -0.221141 -0.128907
free sulfur dioxide  -0.069408 -0.050656
total sulfur dioxide -0.205654 -0.185100
density              -0.496180 -0.174919
pH                    0.205633 -0.057731
sulphates             0.093595  0.251397
alcohol               1.000000  0.476166
quality               0.476166  1.000000
```

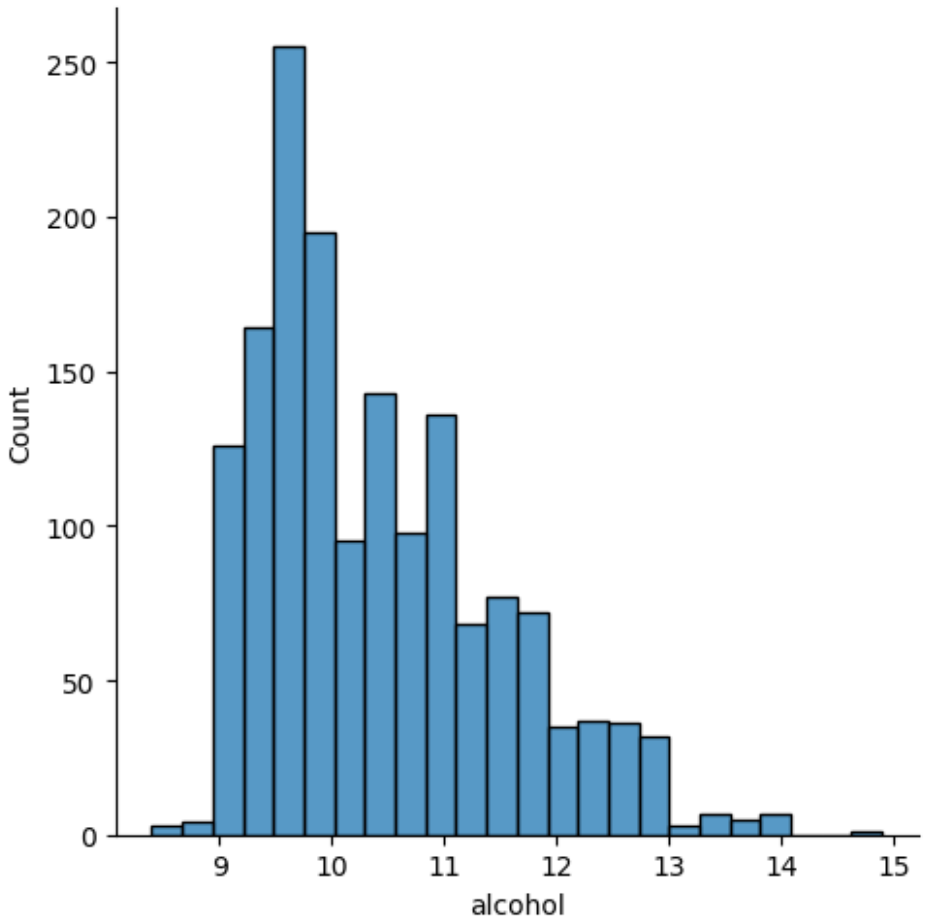
```
import seaborn as sns
sns.heatmap(df.corr() , annot = True   )
```

```
<Axes: >
```



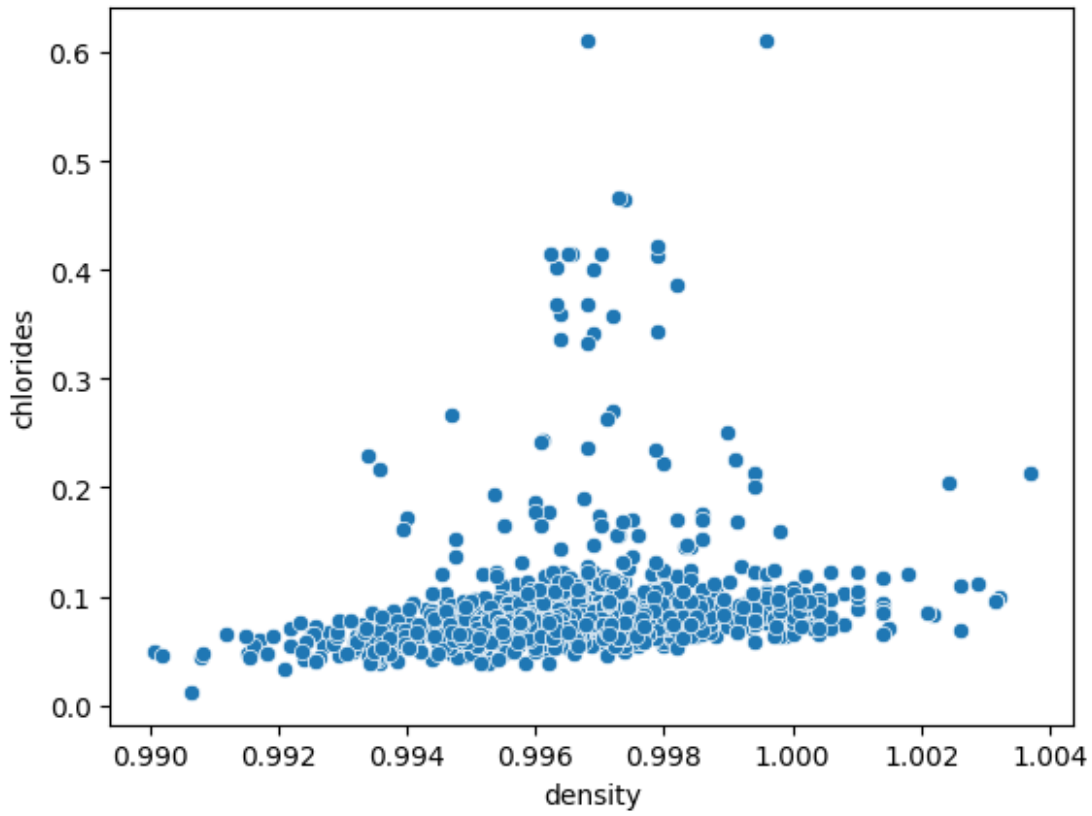

```
sns.displot(df.alcohol)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdcfab50d60>
```



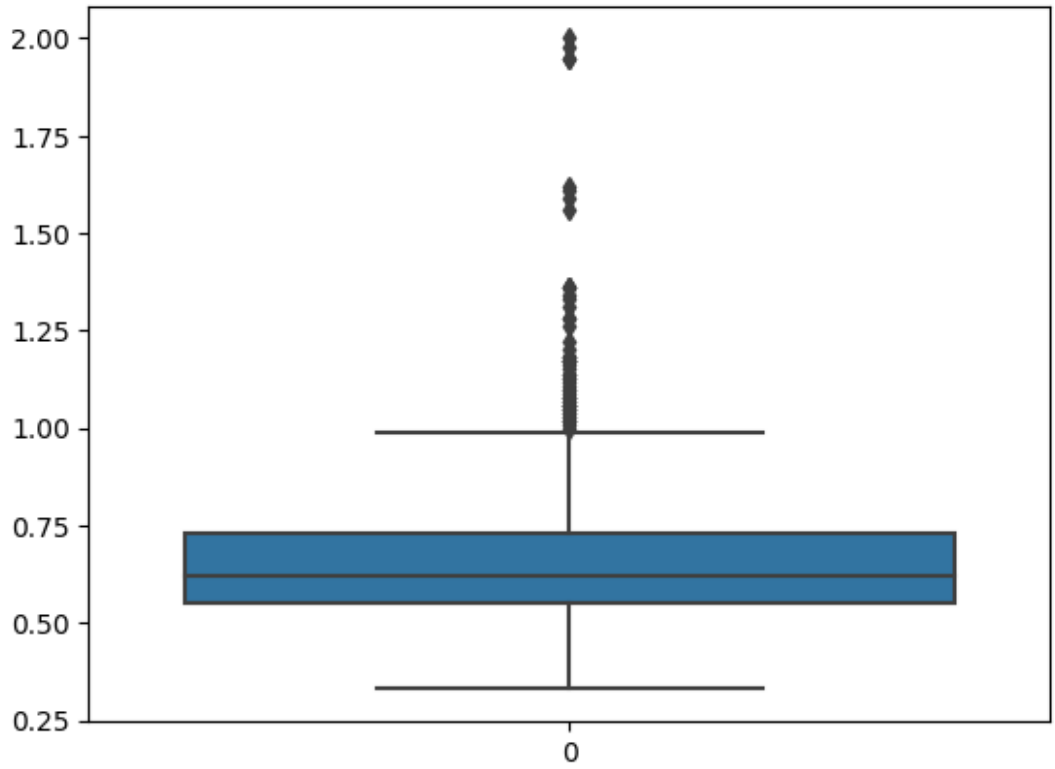

```
sns.scatterplot(x=df.density,y=df.chlorides)
```

```
<Axes: xlabel='density', ylabel='chlorides'>
```




```
sns.boxplot(df.sulphates)
```

```
<Axes: >
```




```
X =df.iloc[:,:-1]
X.head()
```

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

```
y =df['quality']
y.head()
```

```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size =
0.2345,random_state=25)
```

```
x_train.shape
```

```
(1224, 11)
```

```
x_test.shape
```

```
(375, 11)
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
model1 =
DecisionTreeClassifier(max_depth=4,splitter='best',criterion='entropy')
```

```
model1.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
d_y_predict = model1.predict(x_test)
```

```
d_y_predict
```

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

```
d_y_predict_train = model1.predict(x_train)
```

```python
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

print('Testing Accuracy = ', accuracy_score(y_test,d_y_predict))
print('Training Accuracy = ', accuracy_score(y_train,d_y_predict_train))
```

```
Testing Accuracy =  0.5866666666666667
Training Accuracy =  0.6086601307189542
```

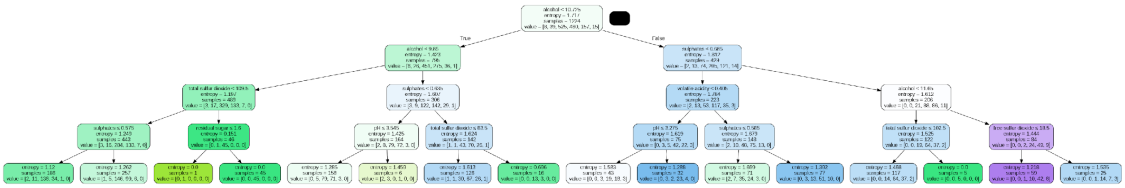
```python
pd.crosstab(y_test,d_y_predict)
```

col_0	4	5	6	7
quality				
3	0	2	0	0
4	0	12	2	0
5	1	133	22	0
6	0	76	78	4
7	0	4	29	9
8	0	1	0	2

```python
print(classification_report(y_test,d_y_predict))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        14
           5       0.58      0.85      0.69       156
           6       0.60      0.49      0.54       158
           7       0.60      0.21      0.32        42
           8       0.00      0.00      0.00         3

    accuracy                           0.59       375
   macro avg       0.30      0.26      0.26       375
weighted avg       0.56      0.59      0.55       375


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import export_graphviz


dot_data =StringIO()
export_graphviz(model1,out_file=dot_data,feature_names= X.columns,
                filled=True,rounded= True,special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



```
from sklearn.ensemble import RandomForestClassifier
model2 =RandomForestClassifier(criterion='entropy')

model2.fit(x_train,y_train)

RandomForestClassifier(criterion='entropy')
```

NEXT

```
r_y_predict = model2.predict(x_test)
r_y_predict_train = model2.predict(x_train)

print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict))
print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train))

Testing Accuracy =  0.7173333333333334
Training Accuracy =  1.0

pd.crosstab(y_test,r_y_predict)

col_0       5    6   7
quality
3           2    0   0
4          11    3   0
5         124   31   1
6          23  122  13
7           3   16  23
8           0    2   1

print(classification_report(y_test,r_y_predict))

              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
```

```
              4       0.00      0.00      0.00        14
              5       0.76      0.79      0.78       156
              6       0.70      0.77      0.73       158
              7       0.61      0.55      0.57        42
              8       0.00      0.00      0.00         3

       accuracy                           0.72       375
      macro avg       0.34      0.35      0.35       375
   weighted avg       0.68      0.72      0.70       375


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.linear_model import LinearRegression
model3 = LinearRegression()
model3.fit(x_train , y_train)
```

```
LinearRegression()
```

```
lr_y_predict = model3.predict(x_test)
lr_y_predict_train = model3.predict(x_train)
```

```
qual = pd.DataFrame( { 'Actual quality' : y_test , 'Predicted quality ' :
lr_y_predict})
```

```
qual
```

```
      Actual quality  Predicted quality
443                7           6.172683
868                6           6.029141
625                5           5.342639
77                 6           5.280204
195                5           4.973205
...              ...                ...
821                7           6.851054
724                4           5.152206
```

```
522                5            5.613748
632                6            5.783782
1270               6            6.897276

[375 rows x 2 columns]
```

```
from sklearn import metrics
```

```
print(metrics.r2_score(y_test,lr_y_predict))
```

```
0.3160722568118429
```

```
print(metrics.r2_score(y_train,lr_y_predict_train))
```

```
0.37108992262715845
```

```
print(metrics.mean_squared_error(y_test,lr_y_predict))
```

```
0.41989394229968163
```

```
print(np.sqrt(metrics.mean_squared_error(y_test,lr_y_predict)))
```

```
0.6479922393822951
```

```
from sklearn.linear_model import LogisticRegression
model4 = LogisticRegression()
```

```
model4.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458
: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
lg_y_pred = model4.predict(x_test)
lg_y_pred
```

```
array([6, 6, 5, 5, 5, 6, 6, 6, 5, 6, 5, 6, 5, 5, 6, 6, 6, 5, 5, 6, 6, 6,
       6, 5, 5, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 5, 6, 6, 5, 5, 5, 5, 6, 6,
       5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 5, 6, 5, 6, 6, 5, 6, 6, 6, 5, 5,
       5, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6, 5, 6, 5, 5, 5, 5, 6, 6, 5, 5, 6,
       6, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 6, 5, 5, 5, 7, 6, 6, 5, 5, 5, 6,
       6, 5, 6, 5, 5, 5, 5, 5, 6, 6, 6, 4, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6,
       6, 6, 5, 5, 5, 6, 5, 5, 5, 6, 5, 5, 6, 5, 5, 6, 6, 6, 5, 6, 5, 5,
       6, 5, 6, 5, 6, 6, 6, 5, 5, 5, 5, 6, 6, 5, 5, 6, 6, 6, 5, 5, 5, 6,
```

```
       6, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, 5, 5, 6, 5, 6,
       5, 5, 5, 6, 5, 5, 6, 5, 5, 6, 6, 6, 5, 5, 6, 5, 6, 6, 6, 6, 5, 5,
       6, 6, 6, 5, 5, 6, 6, 5, 6, 5, 5, 5, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6,
       5, 6, 5, 5, 6, 6, 5, 5, 5, 5, 6, 6, 6, 6, 5, 6, 5, 5, 5, 5, 5, 5,
       6, 6, 6, 6, 5, 6, 5, 5, 5, 5, 5, 5, 6, 5, 5, 6, 5, 6, 6, 6, 6, 5,
       6, 5, 6, 6, 5, 5, 5, 6, 5, 5, 5, 5, 5, 6, 6, 6, 6, 5, 5, 5, 5, 7,
       6, 5, 5, 5, 5, 6, 6, 6, 5, 6, 7, 6, 5, 5, 5, 6, 5, 5, 5, 5, 5, 6,
       6, 6, 6, 5, 5, 5, 5, 6, 5, 5, 6, 6, 6, 6, 5, 6, 5, 5, 5, 5, 5, 6,
       6, 6, 5, 5, 6, 5, 6, 5, 5, 6, 6, 5, 6, 5, 5, 5, 5, 6, 6, 5, 5, 6,
       6])
```

```
accuracy_score(y_test ,  lg_y_pred)
```

```
0.6106666666666667
```

```
confusion_matrix(y_test , lg_y_pred)
```

```
array([[  0,   0,   2,   0,   0,   0],
       [  0,   0,  10,   4,   0,   0],
       [  0,   1, 122,  31,   2,   0],
       [  0,   0,  50, 107,   1,   0],
       [  0,   0,   3,  39,   0,   0],
       [  0,   0,   0,   3,   0,   0]])
```

```
pd.crosstab(y_test  , lg_y_pred)
```

col_0	4	5	6	7
quality				
3	0	2	0	0
4	0	10	4	0
5	1	122	31	2
6	0	50	107	1
7	0	3	39	0
8	0	0	3	0

```
print(classification_report(y_test  , lg_y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        14
           5       0.65      0.78      0.71       156
           6       0.58      0.68      0.63       158
           7       0.00      0.00      0.00        42
           8       0.00      0.00      0.00         3

    accuracy                           0.61       375
   macro avg       0.21      0.24      0.22       375
weighted avg       0.52      0.61      0.56       375
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
probability  = model4.predict_proba(x_test)[:,1]
probability
```

```
array([0.02231348, 0.04163394, 0.08519507, 0.05083156, 0.00490805,
       0.0193458 , 0.01256215, 0.05289513, 0.02781566, 0.01464755,
       0.03931959, 0.0107028 , 0.02987398, 0.03434318, 0.01468408,
       0.02393854, 0.01709038, 0.01847209, 0.01306171, 0.02298114,
       0.02619998, 0.04091116, 0.02199939, 0.01284045, 0.02405573,
       0.00785142, 0.02713541, 0.03233056, 0.02981752, 0.01433764,
       0.06836174, 0.0186473 , 0.01066855, 0.05500996, 0.01619648,
       0.01267665, 0.01578444, 0.02282645, 0.02461835, 0.01978253,
       0.0366563 , 0.00363948, 0.03575345, 0.04148333, 0.00707848,
       0.00969808, 0.02940425, 0.06298075, 0.00617853, 0.00585006,
       0.03995384, 0.02742406, 0.04740601, 0.06184464, 0.03449721,
       0.0141726 , 0.02072323, 0.0093657 , 0.04654918, 0.02504415,
       0.19823629, 0.02288114, 0.03971682, 0.02271076, 0.01426226,
       0.0184628 , 0.02519693, 0.04180825, 0.03298282, 0.03448643,
       0.01817787, 0.03430499, 0.00371764, 0.00331719, 0.04773681,
       0.01445913, 0.25958839, 0.02240472, 0.01998612, 0.0083854 ,
       0.00468811, 0.02310817, 0.00951864, 0.00424974, 0.01706115,
       0.03465065, 0.02413885, 0.00796555, 0.03575345, 0.02418073,
       0.02603966, 0.04610266, 0.01355445, 0.00806668, 0.08603429,
       0.01993684, 0.15239108, 0.02791904, 0.00869844, 0.08992853,
       0.00129601, 0.01554423, 0.01371833, 0.00274211, 0.05441823,
       0.04367245, 0.02878639, 0.02582325, 0.04713568, 0.02745255,
       0.01239106, 0.02392153, 0.0140506 , 0.02836111, 0.00273512,
       0.02201674, 0.04002607, 0.03318639, 0.02053827, 0.03047365,
       0.03303124, 0.68922827, 0.01217775, 0.02117164, 0.01833082,
       0.04086002, 0.00584256, 0.01013715, 0.02880882, 0.02711344,
       0.01893144, 0.05205817, 0.02254521, 0.01161684, 0.05260757,
       0.01383345, 0.21505973, 0.04839655, 0.01411707, 0.05561837,
       0.01286617, 0.05404468, 0.02219081, 0.01462808, 0.03717614,
       0.03958877, 0.01387729, 0.02272274, 0.05528297, 0.0707622 ,
       0.01789632, 0.02126258, 0.01967015, 0.01286476, 0.08757728,
```

0.00810907, 0.06652192, 0.0224268 , 0.06754285, 0.01148671,
0.00948688, 0.00258715, 0.03756595, 0.03000267, 0.01241779,
0.01042694, 0.00656652, 0.03977784, 0.07695126, 0.01902549,
0.01501748, 0.0205284 , 0.01929558, 0.03310709, 0.00361206,
0.03418034, 0.04904556, 0.02493929, 0.01874508, 0.03865356,
0.03438198, 0.02647355, 0.01888033, 0.0265591 , 0.07917147,
0.01806826, 0.01693869, 0.01311486, 0.02970983, 0.02702479,
0.00776865, 0.0260961 , 0.0182592 , 0.00638948, 0.00271141,
0.03228054, 0.04512922, 0.03141222, 0.01340568, 0.04955362,
0.01173407, 0.00765801, 0.00767627, 0.02063833, 0.0354255 ,
0.02403632, 0.02600112, 0.00818796, 0.05913576, 0.02411407,
0.01300767, 0.02291231, 0.00375394, 0.03328894, 0.03614571,
0.02786684, 0.01648051, 0.02738449, 0.02614384, 0.00803449,
0.02085605, 0.15377695, 0.01029408, 0.06473149, 0.02467247,
0.03019743, 0.03559981, 0.03135863, 0.02644586, 0.00435409,
0.00575577, 0.04662575, 0.00856089, 0.03598631, 0.03452478,
0.05549676, 0.04745765, 0.05605744, 0.03443901, 0.02355309,
0.03882681, 0.0161406 , 0.00546725, 0.0509455 , 0.02545856,
0.00887054, 0.02979742, 0.02263628, 0.03382728, 0.03877168,
0.02000541, 0.01321581, 0.03124845, 0.01485985, 0.04748688,
0.01652912, 0.02238656, 0.02322026, 0.01755076, 0.05841008,
0.02262814, 0.01682981, 0.00592624, 0.01209896, 0.01786638,
0.00921909, 0.04818858, 0.0693027 , 0.01340651, 0.00421636,
0.03763882, 0.01738477, 0.04006769, 0.0207194 , 0.02333807,
0.01153046, 0.02235526, 0.00466316, 0.03094452, 0.01764527,
0.01911233, 0.03784202, 0.02541535, 0.05042481, 0.01960221,
0.04661615, 0.01649633, 0.0350239 , 0.0174772 , 0.01985802,
0.01149449, 0.00428788, 0.04610148, 0.06126463, 0.02584102,
0.04661615, 0.02836242, 0.21505973, 0.01304011, 0.01312191,
0.00657444, 0.02479886, 0.00921909, 0.01426012, 0.08678045,
0.012808 , 0.01553519, 0.00784356, 0.03019743, 0.01257151,
0.01820413, 0.01728637, 0.05909676, 0.0184561 , 0.04059881,
0.04043128, 0.02357973, 0.01847568, 0.01657043, 0.02786684,
0.03112493, 0.02788704, 0.02614478, 0.07376728, 0.014298 ,
0.00746905, 0.03159653, 0.01252244, 0.03864074, 0.04397804,
0.02887132, 0.05549676, 0.09493548, 0.02529333, 0.0226742 ,
0.18980981, 0.00445433, 0.14474373, 0.01284395, 0.08519507,
0.04278993, 0.01131672, 0.0073023 , 0.03830643, 0.00554878,
0.01308248, 0.02154052, 0.04165767, 0.03118349, 0.00910053,
0.03187074, 0.04858912, 0.01931963, 0.0638396 , 0.01237313,
0.02037462, 0.01250526, 0.03224592, 0.04745765, 0.043191 ,
0.03885703, 0.01485499, 0.05314937, 0.01597418, 0.01482451,
0.03644874, 0.01257615, 0.03145768, 0.02008216, 0.03614414,
0.04299545, 0.08733117, 0.00323518, 0.03650397, 0.0230185])