1. Use your own dataset (Heart.csv is acceptable), create a train and a test set, and build 2 models: Logistic Regression and Decision Tree (shallow). Compare the test results using classification\_report and confusion\_matrix. Explain which algorithm is optimal

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
```

## I used a dataset from UCI: Wine Quality (Red wine)

(https://archive.ics.uci.edu/dataset/186/wine+quality\_(https://archive.ics.uci.edu/dataset/186/wine+quality\_))

- This data shows scores of wine quality(from 1 to 10) with 11 related features
- Good wine : score >= 6, normal wine : score < 6

## 1-A. Preprocessing

```
In [2]: # read the dataset as 'df'
df = pd.read_csv("winequality-red.csv")
df.head()
```

#### Out[2]:

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

- · As variable 'quality' has values from 1 to 10, transforming the values into binary values is necessary
- values from 6 to 10 are transformed into value 1(good wine), and values from 1 to 5 are transformed into value 0(normal wine)

```
In [3]: # transform values from 6 to 10 into value 1
    df['quality'] = df.quality.between(6,10).astype(int)
    df.head()
```

#### Out[3]:

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

## 1-B. Logistic Regression

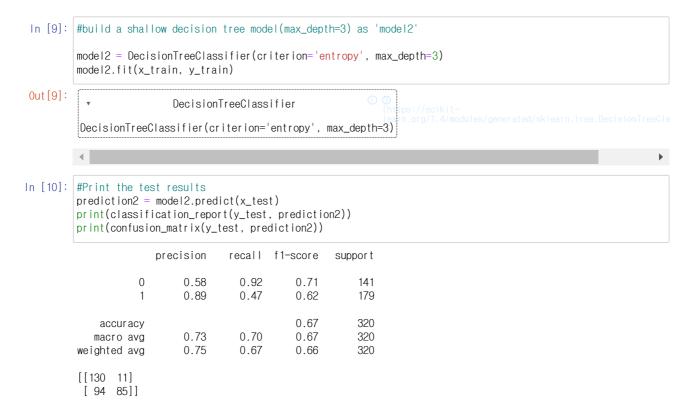
[ 46 133]]

```
In [6]: |#generate logistic regression model as 'model1'
                    model1 = LogisticRegression()
                    model1.fit(x_train, y_train)
                    C:\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Users\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Union\Uni
                     fgs 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 (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 (https://scikit-learn.
                    org/stable/modules/linear_model.html#logistic-regression)
                         n_iter_i = _check_optimize_result(
Out[6]:

    LogisticRegression

                     LogisticRegression()
In [7]: #As the warning sign indicates total number of iterations reached limit,
                    #1 increased maximum number of iterations from 100(defalut) to 1000
                    model1 = LogisticRegression(max_iter=1000)
                    model1.fit(x_train, y_train)
Out [7]:
                                      LogisticRegression
                     LogisticRegression(max_iter=1000)
In [8]: #Print the test results
                    prediction1 = model1.predict(x_test)
                    print(classification_report(y_test, prediction1))
                    print(confusion_matrix(y_test, prediction1))
                                                      precision
                                                                                     recall f1-score
                                                                                                                                  support
                                               0
                                                                  0.70
                                                                                          0.74
                                                                                                                 0.72
                                                                                                                                             141
                                                                  0.79
                                                                                                                 0.76
                                                                                          0.74
                                                                                                                                             179
                                                                                                                 0.74
                                                                                                                                            320
                              accuracy
                                                                  0.74
                                                                                          0.74
                                                                                                                  0.74
                                                                                                                                            320
                            macro avg
                                                                  0.75
                                                                                                                 0.74
                                                                                                                                            320
                    weighted avg
                                                                                          0.74
                     [[105 36]
```

#### 1-C. Decision Tree

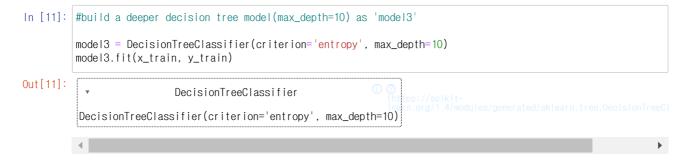


#### Result: Comparing cell 8 and 10, Logistic Regression model is more optimal.

• Logistic Regression model shows higher f1-score in predicting good wine (value 1, 0.76 > 0.62) and normal wine(value 0, 0.72 > 0.71)

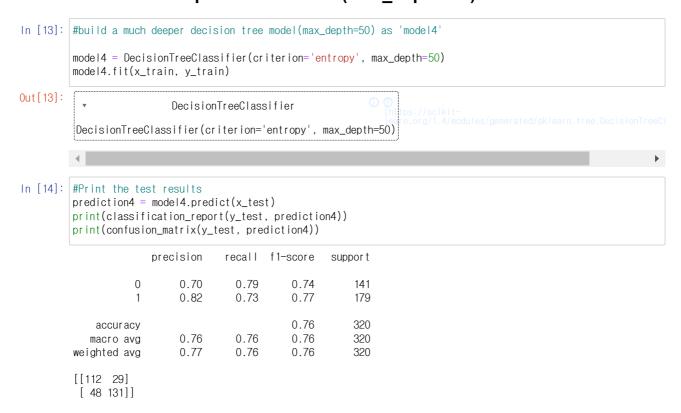
### 2. Repeat 1. but let the Decision Tree be much deeper to allow overfitting. Compare the two models' test results again, and explain which is optimal

## 2-A. Deeper Decision Tree(max depth=10)



```
In [12]: #Print the test results
         prediction3 = model3.predict(x_test)
         print(classification_report(y_test, prediction3))
        print(confusion_matrix(y_test, prediction3))
                       precision
                                   recall f1-score support
                    0
                            0.71
                                      0.75
                                                0.73
                                                           141
                            0.79
                    1
                                      0.75
                                                0.77
                                                           179
                                                0.75
             accur acy
                                                           320
                            0.75
                                      0.75
                                               0.75
                                                           320
            macro avg
         weighted avg
                           0.76
                                     0.75
                                               0.75
                                                           320
         [[106 35]
          [ 44 135]]
```

## 2-B. Much Deeper Decision Tree(max\_depth=50)



# Result : Comparing the cell 8, 12, and 14, deeper Decision Tree model outperforms Logistic Regression model

- both Decision Tree model shows higher f1-score than Logistic Regression model
- · f1-score improved as max depth of Decision Tree increased