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

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A Num
Py version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected v
ersion 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [ ]:
```
df = pd.read_csv("/kaggle/input/heart-attack-analysis-prediction-dataset/heart.c
df
```

Out[ ]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	..
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 14 columns

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
       'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],
      dtype='object')
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

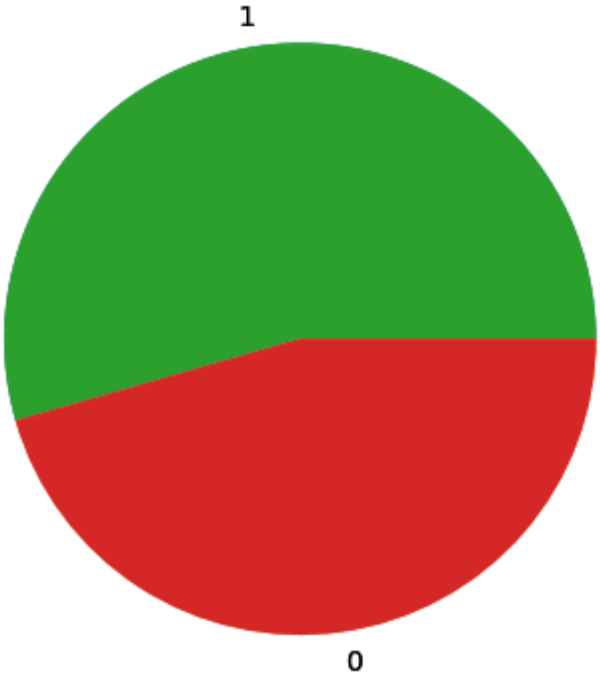
	age	sex	cp	trtbps	chol	fbs	reste
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528(
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525{
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000(
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000(
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000(

In [ ]:
```
# Showing the frequency of categorical data in each feature of the dataset
for i in df.columns:
    if len(df[i].value_counts()) <= 10:
        print(i,":")
        print(df[i].value_counts())
        print()
```

```
sex :
1    207
0     96
Name: sex, dtype: int64

cp :
0    143
2     87
1     50
3     23
Name: cp, dtype: int64

fbs :
0    258
1     45
Name: fbs, dtype: int64

restecg :
1    152
0    147
2      4
Name: restecg, dtype: int64

exng :
0    204
1     99
Name: exng, dtype: int64

slp :
2    142
1    140
0     21
Name: slp, dtype: int64

caa :
0    175
1     65
2     38
3     20
4      5
Name: caa, dtype: int64

thall :
2    166
3    117
1     18
0      2
Name: thall, dtype: int64

output :
1    165
0    138
Name: output, dtype: int64
```

In [ ]:
```
plt.pie(df['output'].value_counts(), labels=pd.unique(df['output']))
plt.pie(df['output'].value_counts(), labels=pd.unique(df['output']))
```

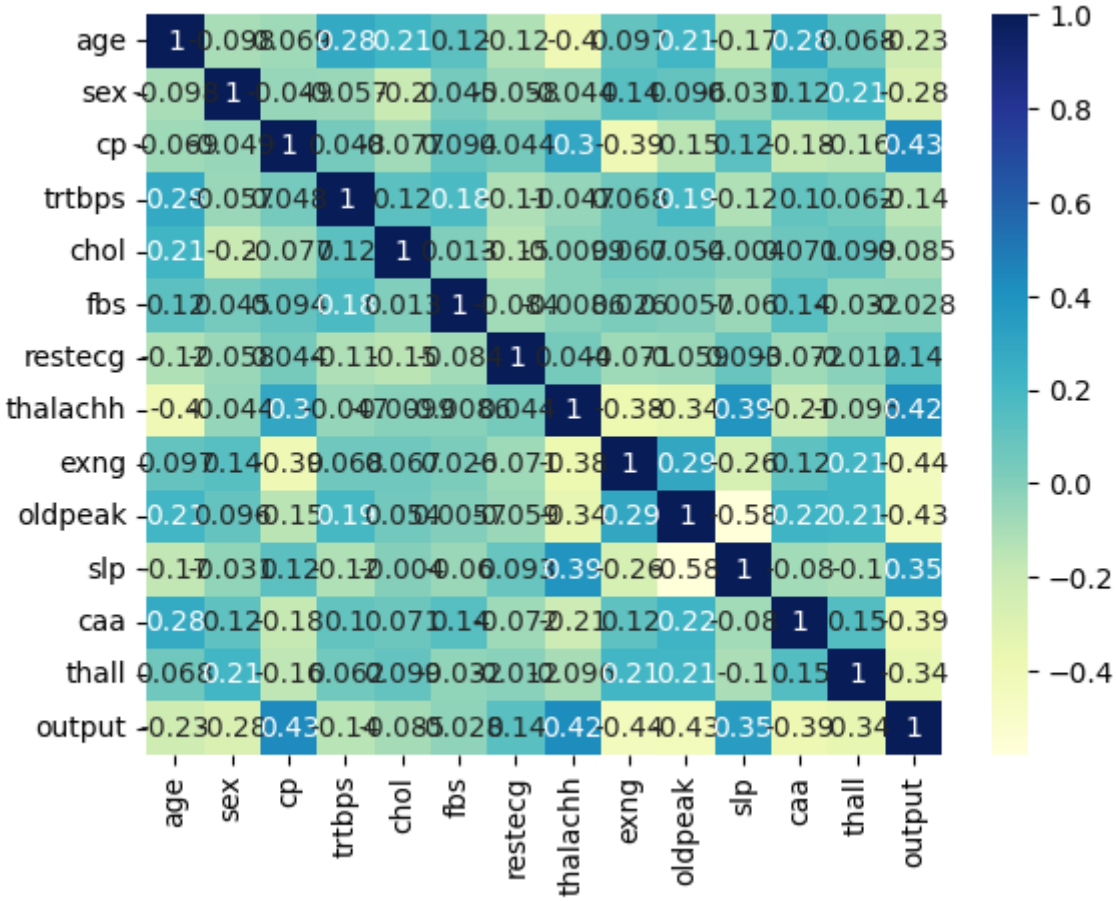
Out[ ]:
```
([<matplotlib.patches.Wedge at 0x7a2ed0aff580>,
  <matplotlib.patches.Wedge at 0x7a2ed0e70df0>],
 [Text(-0.1534669293828939, 1.0892418930548835, '1'),
  Text(0.1534670313650796, -1.0892418786862677, '0')])
```



In [ ]:
```
y = df['output']
X = df.drop(["output"], axis=1)
X,y
```

Out[ ]:
```
(     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
0     63    1   3     145   233    1        0       150     0      2.3    0
1     37    1   2     130   250    0        1       187     0      3.5    0
2     41    0   1     130   204    0        0       172     0      1.4    2
3     56    1   1     120   236    0        1       178     0      0.8    2
4     57    0   0     120   354    0        1       163     1      0.6    2
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
298   57    0   0     140   241    0        1       123     1      0.2    1
299   45    1   3     110   264    0        1       132     0      1.2    1
300   68    1   0     144   193    1        1       141     0      3.4    1
301   57    1   0     130   131    0        1       115     1      1.2    1
302   57    0   1     130   236    0        0       174     0      0.0    1

     caa  thall
0      0      1
1      0      2
2      0      2
3      0      2
4      0      2
..   ...    ...
298    0      3
299    0      3
300    2      3
301    1      3
302    1      2

[303 rows x 13 columns],
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: output, Length: 303, dtype: int64)
```

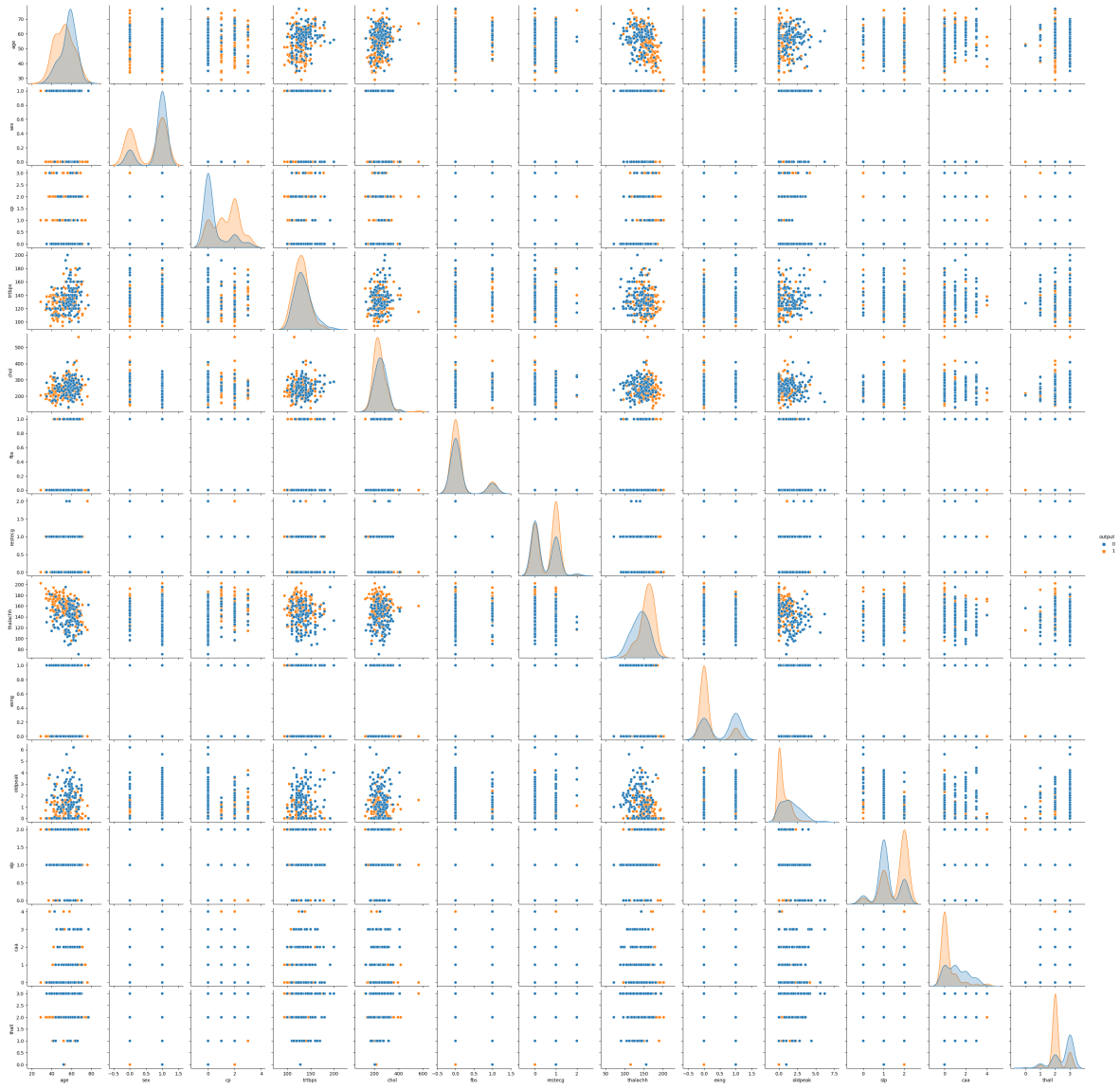
In [ ]:
```
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```



The correlation above shows that there aren't any highly correlated features in the dataset

In [ ]:
```
sns.pairplot(df,hue="output")
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7a2ece66bb50>
```



In [ ]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_cols = X.columns
X = scaler.fit_transform(X)

X = pd.DataFrame(X,columns = X_cols)
X
```

Out[ ]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpea
0	0.708333	1.0	1.000000	0.481132	0.244292	1.0	0.0	0.603053	0.0	0.37096
1	0.166667	1.0	0.666667	0.339623	0.283105	0.0	0.5	0.885496	0.0	0.56451
2	0.250000	0.0	0.333333	0.339623	0.178082	0.0	0.0	0.770992	0.0	0.22580
3	0.562500	1.0	0.333333	0.245283	0.251142	0.0	0.5	0.816794	0.0	0.12903
4	0.583333	0.0	0.000000	0.245283	0.520548	0.0	0.5	0.702290	1.0	0.09677
...	...	...	...	...	...	...	...	...	...	
298	0.583333	0.0	0.000000	0.433962	0.262557	0.0	0.5	0.396947	1.0	0.03225
299	0.333333	1.0	1.000000	0.150943	0.315068	0.0	0.5	0.465649	0.0	0.19354
300	0.812500	1.0	0.000000	0.471698	0.152968	1.0	0.5	0.534351	0.0	0.54838
301	0.583333	1.0	0.000000	0.339623	0.011416	0.0	0.5	0.335878	1.0	0.19354
302	0.583333	0.0	0.333333	0.339623	0.251142	0.0	0.0	0.786260	0.0	0.00000

303 rows × 13 columns

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X.to_numpy(),y.to_numpy(),test_
X_train.shape[0]
```

Out[ ]: 212

In [ ]:
```
# Using random sampling technique to select different sizes of training data
from random import sample
perc = 100
samples= int(X_train.shape[0]*(perc/100))
workflow_indices = sample(list(np.arange(0,X_train.shape[0])),samples)
X_work = X_train[workflow_indices]
y_work = y_train[workflow_indices]

samples
```

Out[ ]: 212

In [ ]:
```
class KNN_classifier:
    def __init__(self,k_neighbours):
        self.k_neighbours=k_neighbours

    def minkowski(self,x_tr,x_tt,p=2):  # P=2 refers to euclidean distance
        return (sum([(abs(x_tr[i]-x_tt[i]))**p for i in range(len(x_tr))]))**(1/

    def fit(self,X,y):
        self.X = X
        self.y = y

    def predict(self,x_test,p=2):
        distances = []
        for train_id in range(len(self.X)):
```

```
            try:
                if len(x_test) != len(self.X[train_id]):
                    raise Exception
                distances.append([self.minkowski(x_test,self.X[train_id],p), sel
            except:
                print("Length of train data and prediction data not matching")
                break

        distances.sort(key = lambda tup: tup[0])
        freq = np.bincount(np.array(distances, dtype=np.int64)[:self.k_neighbour
        return(np.argmax(freq))
```

In [ ]:
```
from time import process_time

for k in range(3,7):
    t1 = process_time()
    classifier = KNN_classifier(k)
    classifier.fit(X_work,y_work)

    # Predicting the Test set results
    y_test_pred = []
    for i in X_test:
        y_test_pred.append(classifier.predict(i))

    t2 = process_time()

    print(f"Time taken for training(train data) and predicting(test data) {k}:",

#     from sklearn.metrics import classification_report

#     acc = sum([1 if y_test_pred[i] == y_test[i] else 0 for i in range(len(y_te
#     print("Accuracy:",acc)
#     print(classification_report(y_test,y_test_pred))
```

```
Time taken for training(train data) and predicting(test data) 3: 0.36430781500000
364
Time taken for training(train data) and predicting(test data) 4: 0.34091543600000
307
Time taken for training(train data) and predicting(test data) 5: 0.34207887899999
89
Time taken for training(train data) and predicting(test data) 6: 0.34221773699999
86
```

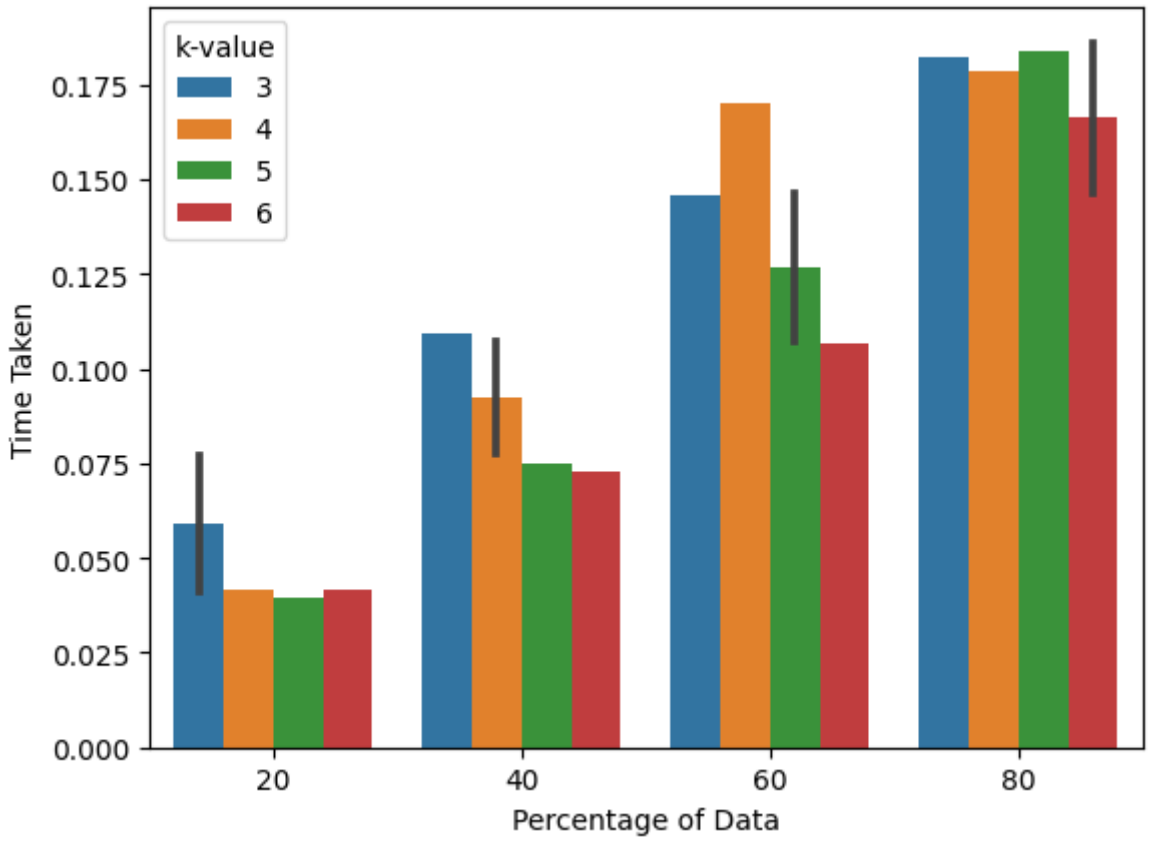
In [ ]:
```
time_analysis_k = [0.041011325,0.041398121,0.039613125,0.041703245,
0.077000163,0.077383058,0.074809444,0.072773152,
0.109130311,0.107329887,0.107174437,0.106888516,
0.145766027,0.169836267,0.146234051,0.146342354,
0.182187114,0.178715758,0.183862835,0.186098923]

perc = [20,20,20,20,20,40,40,40,40,40,60,60,60,60,60,80,80,80,80,80,100,100,100,
k_vals = [3,4,5,6]*5
# dt = np.concatenate([perc,time_analysis_k,],axis=1)
time_analysis = pd.DataFrame(list(zip(perc, time_analysis_k, k_vals)), columns=[
time_analysis
```

Out[ ]:

	perc	time	k-value
0	20	0.041011	3
1	20	0.041398	4
2	20	0.039613	5
3	20	0.041703	6
4	20	0.077000	3
5	40	0.077383	4
6	40	0.074809	5
7	40	0.072773	6
8	40	0.109130	3
9	40	0.107330	4
10	60	0.107174	5
11	60	0.106889	6
12	60	0.145766	3
13	60	0.169836	4
14	60	0.146234	5
15	80	0.146342	6
16	80	0.182187	3
17	80	0.178716	4
18	80	0.183863	5
19	80	0.186099	6

In [ ]:
```
sns.barplot(x = 'perc',y = 'time', hue="k-value", data=time_analysis)
plt.xlabel("Percentage of Data")
plt.ylabel("Time Taken")
```

Out[ ]:
```
Text(0, 0.5, 'Time Taken')
```



In [ ]:
```
for k in range(3,7):
    classifier = KNN_classifier(k)
    classifier.fit(X_train,y_train)

    # Predicting the Test set results
    y_test_pred = []
    for i in X_test:
        y_test_pred.append(classifier.predict(i))

    from sklearn.metrics import classification_report
    from sklearn.metrics import precision_recall_fscore_support

    acc = sum([1 if y_test_pred[i] == y_test[i] else 0 for i in range(len(y_test
    print("Accuracy:",acc)

    print(classification_report(y_test,y_test_pred))

    print(precision_recall_fscore_support(y_test,y_test_pred, average='binary'))
```

```
Accuracy: 0.7912087912087912
              precision    recall  f1-score   support

           0       0.81      0.71      0.76        42
           1       0.78      0.86      0.82        49

    accuracy                           0.79        91
   macro avg       0.79      0.79      0.79        91
weighted avg       0.79      0.79      0.79        91

(0.7777777777777778, 0.8571428571428571, 0.8155339805825242, None)
Accuracy: 0.7912087912087912
              precision    recall  f1-score   support

           0       0.76      0.81      0.78        42
           1       0.83      0.78      0.80        49

    accuracy                           0.79        91
   macro avg       0.79      0.79      0.79        91
weighted avg       0.79      0.79      0.79        91

(0.8260869565217391, 0.7755102040816326, 0.8, None)
Accuracy: 0.7912087912087912
              precision    recall  f1-score   support

           0       0.79      0.74      0.77        42
           1       0.79      0.84      0.81        49

    accuracy                           0.79        91
   macro avg       0.79      0.79      0.79        91
weighted avg       0.79      0.79      0.79        91

(0.7884615384615384, 0.8367346938775511, 0.8118811881188118, None)
Accuracy: 0.8021978021978022
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        42
           1       0.82      0.82      0.82        49

    accuracy                           0.80        91
   macro avg       0.80      0.80      0.80        91
weighted avg       0.80      0.80      0.80        91

(0.8163265306122449, 0.8163265306122449, 0.8163265306122449, None)
```

In [ ]: