genetic-algorithm

December 23, 2023

[423]:
```
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.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score
from deap import base, creator, tools, algorithms
```

[424]:
```
data = pd.read_csv("/kaggle/input/breast-cancer-wisconsin-data/data.csv")
data.head()
```

[424]:
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  …          17.33           184.60      2019.0            0.1622
1  …          23.41           158.80      1956.0            0.1238
2  …          25.53           152.50      1709.0            0.1444
3  …          26.50            98.87       567.7            0.2098
4  …          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
```

```
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[425]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

[426]:
```
data.describe()
```

[426]:
```
                 id  radius_mean  texture_mean  perimeter_mean    area_mean  \
count  5.690000e+02   569.000000    569.000000      569.000000   569.000000
mean   3.037183e+07    14.127292     19.289649       91.969033   654.889104
std    1.250206e+08     3.524049      4.301036       24.298981   351.914129
min    8.670000e+03     6.981000      9.710000       43.790000   143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000   420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000   551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000   782.700000
max    9.113205e+08    28.110000     39.280000      188.500000  2501.000000

       smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
count       569.000000        569.000000      569.000000           569.000000
mean          0.096360          0.104341        0.088799             0.048919
std           0.014064          0.052813        0.079720             0.038803
min           0.052630          0.019380        0.000000             0.000000
25%           0.086370          0.064920        0.029560             0.020310
50%           0.095870          0.092630        0.061540             0.033500
75%           0.105300          0.130400        0.130700             0.074000
max           0.163400          0.345400        0.426800             0.201200

       symmetry_mean  ...  texture_worst  perimeter_worst   area_worst  \
count     569.000000  ...     569.000000       569.000000   569.000000
mean        0.181162  ...      25.677223       107.261213   880.583128
std         0.027414  ...       6.146258        33.602542   569.356993
min         0.106000  ...      12.020000        50.410000   185.200000
25%         0.161900  ...      21.080000        84.110000   515.300000
50%         0.179200  ...      25.410000        97.660000   686.500000
75%         0.195700  ...      29.720000       125.400000  1084.000000
max         0.304000  ...      49.540000       251.200000  4254.000000

       smoothness_worst  compactness_worst  concavity_worst  \
count        569.000000         569.000000       569.000000
mean           0.132369           0.254265         0.272188
std            0.022832           0.157336         0.208624
min            0.071170           0.027290         0.000000
25%            0.116600           0.147200         0.114500
```

```
50%            0.131300          0.211900         0.226700
75%            0.146000          0.339100         0.382900
max            0.222600          1.058000         1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst  \
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

       Unnamed: 32
count          0.0
mean           NaN
std            NaN
min            NaN
25%            NaN
50%            NaN
75%            NaN
max            NaN

[8 rows x 32 columns]
```

[427]:

```
# Missing values
data.isna().sum()
```

[427]:

```
id                            0
diagnosis                     0
radius_mean                   0
texture_mean                  0
perimeter_mean                0
area_mean                     0
smoothness_mean               0
compactness_mean              0
concavity_mean                0
concave points_mean           0
symmetry_mean                 0
fractal_dimension_mean        0
radius_se                     0
texture_se                    0
perimeter_se                  0
area_se                       0
smoothness_se                 0
compactness_se                0
```

```
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

[428]:

```
# Drop the id and 'Unnamed: 32' column as it contains only NaN values
data = data.drop(columns=['id','Unnamed: 32'], axis=1)
data.head()
```

[428]:

```
  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  …         25.38          17.33           184.60
1         0.1812  …         24.99          23.41           158.80
2         0.2069  …         23.57          25.53           152.50
3         0.2597  …         14.91          26.50            98.87
4         0.1809  …         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
```

```
4      1575.0           0.1374           0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 31 columns]
```

[429]:
```
data.shape
```

[429]:
```
(569, 31)
```

[430]:
```
# Duplicated observations
data.duplicated().sum()
```

[430]:
```
0
```

[431]:
```
# Map diagnosis to 0 (Benign) and 1 (Malignant)
data['diagnosis'] = data['diagnosis'].map({'M': 1, 'B': 0})
```

[432]:
```
# Separate features and target variable
X = data.drop('diagnosis', axis=1)
y = data['diagnosis']
```

[433]:
```
# Split the data into training, validation, and testing sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,␣
 ↪random_state=42)
```

1 Define the genetic algorithm functions

[434]:
```
def evaluate(individual, X_train, y_train, X_val, y_val):
    # Concatenate the arrays in the individual list
    combined_individual = np.concatenate([np.expand_dims(arr, axis=0) if␣
 ↪len(arr.shape) == 1 else arr for arr in individual])

    # Reshape to match the number of columns in X
    combined_individual = combined_individual.reshape(-1, X_train.shape[1])

    # Create a mask based on the individual's genes
    mask = np.array(combined_individual, dtype=bool)
    # Select features using the mask
    X_train_selected = X_train.iloc[:, mask]
    X_val_selected = X_val.iloc[:, mask]

    # Train a RandomForestClassifier
    clf = RandomForestClassifier(random_state=42)
    clf.fit(X_train_selected, y_train)

    # Make predictions on the validation set
    y_val_pred = clf.predict(X_val_selected)

    # Calculate accuracy on the validation set
    accuracy_val = accuracy_score(y_val, y_val_pred)

    return accuracy_val,
```

2 Genetic Algorithm Setup

[435]:

```
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()
toolbox.register("attr_bool", np.random.choice, [0, 1], size=X.shape[1])
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.
  ↪attr_bool, n=X.shape[1])
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

toolbox.register("evaluate", evaluate, X_train=X_train, y_train=y_train,␣
  ↪X_val=X_val, y_val=y_val)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
```

```
/opt/conda/lib/python3.10/site-packages/deap/creator.py:185: RuntimeWarning: A
class named 'FitnessMax' has already been created and it will be overwritten.
Consider deleting previous creation of that class or rename it.
  warnings.warn("A class named '{0}' has already been created and it "
/opt/conda/lib/python3.10/site-packages/deap/creator.py:185: RuntimeWarning: A
class named 'Individual' has already been created and it will be overwritten.
Consider deleting previous creation of that class or rename it.
  warnings.warn("A class named '{0}' has already been created and it "
```

```
gen     nevals  avg             min             max
0       10      0.964706        0.964706        0.964706
1       2       0.964706        0.964706        0.964706
2       2       0.964706        0.964706        0.964706
3       2       0.964706        0.964706        0.964706
```

```
4        2        0.964706        0.964706        0.964706
5        2        0.964706        0.964706        0.964706
```

3 Genetic Algorithm

[ ]:

```python
population_size = 10
crossover_prob = 0.8
mutation_prob = 0.2
generations = 5
```

[ ]:

```python
population = toolbox.population(n=population_size)
```

[ ]:

```python
# Track statistics during the evolution
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", np.mean)
stats.register("min", np.min)
stats.register("max", np.max)
```

4 Execute the genetic algorithm

[ ]:

```python
population, logbook = algorithms.eaMuPlusLambda(population, toolbox,␣
  ↪mu=population_size, lambda_=2, cxpb=crossover_prob, mutpb=mutation_prob,␣
  ↪ngen=generations, stats=stats, halloffame=None, verbose=True)
```

[436]:

```python
# Extract the best individual
best_individual = tools.selBest(population, k=1)[0]
```

[437]:

```python
# Evaluate the best individual on the test set
best_mask = np.array(best_individual, dtype=bool)
X_test_selected = X_test.iloc[:, best_mask]
```

[438]:

```python
# Train a RandomForestClassifier with the selected features
best_clf = RandomForestClassifier(random_state=42)
best_clf.fit(X_train.iloc[:, best_mask], y_train)
```

[438]:

```
RandomForestClassifier(random_state=42)
```

[439]:

```python
# Make predictions on the test set
y_pred_test = best_clf.predict(X_test_selected)
```

[441]:

```python
# Evaluate performance on the test set
from sklearn.metrics import accuracy_score, confusion_matrix,␣
  ↪classification_report

accuracy_test = accuracy_score(y_test, y_pred_test)
```

```
conf_matrix = confusion_matrix(y_test, y_pred_test)
classification_rep = classification_report(y_test, y_pred_test)
```

[442]:
```
# Display results
print(f"Best Individual: {best_individual}")
print("---------------------------------------------------------")
print(f"Accuracy : {accuracy_test}")
print("---------------------------------------------------------")
print("Classification Report:")
print(classification_rep)
```

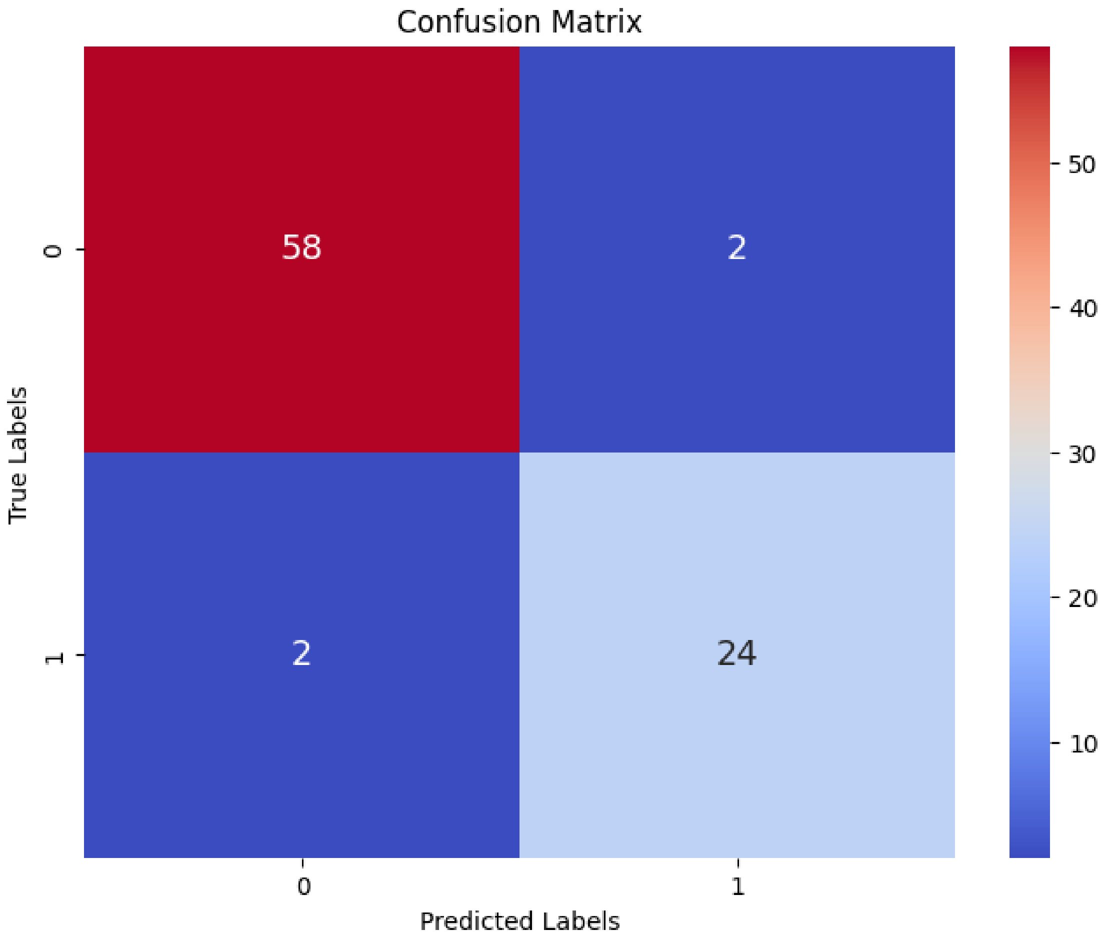
```
Best Individual: [array([1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0,
1, 1, 1, 1,
       0, 1, 1, 1, 1, 0, 1, 1]), array([1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 0]), array([1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 0]), array([1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0,
0, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 0]), array([0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 0]), array([0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0]), array([0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
0, 1, 1, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0]), array([1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 1]), array([1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
0, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 1, 1]), array([0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 0]), array([0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,
0, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1]), array([1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
0, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 1]), array([0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
0, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 0]), array([0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 0]), array([0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 1]), array([1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
0, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 0]), array([1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1]), array([1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1]), array([0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0]), array([1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
0, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 1]), array([1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 0, 1, 1]), array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 1]), array([1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0]), array([1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 0]), array([0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 1, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0]), array([0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
0, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1]), array([1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
1, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 0, 1]), array([0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0,
0, 1, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 0, 0, 1]), array([1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0]), array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 1])]
---------------------------------------------------------
Accuracy : 0.9534883720930233
---------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        60
           1       0.92      0.92      0.92        26

    accuracy                           0.95        86
   macro avg       0.94      0.94      0.94        86
weighted avg       0.95      0.95      0.95        86
```

[443]:

```
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[58  2]
 [ 2 24]]
```

```
[445]: # Plot confusion matrix as a heatmap
       plt.figure(figsize=(8, 6))
       sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='coolwarm', cbar=True,␣
        ↪annot_kws={"size": 14})
       plt.title('Confusion Matrix')
       plt.xlabel('Predicted Labels')
       plt.ylabel('True Labels')
       plt.show()
```


Confusion Matrix
True Labels
Predicted Labels