In [1]:

```
import pandas as pd
import numpy as np
from sklearn import *
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import f1_score
```

In [2]:

```
df = pd.read_csv("titanic_train.csv")
```

In [3]:

```
df.dtypes
```

Out[3]:

```
passenger_id      int64
pclass            int64
name             object
sex              object
age             float64
sibsp             int64
parch             int64
ticket           object
fare            float64
cabin            object
embarked         object
boat             object
body            float64
home.dest        object
survived          int64
dtype: object
```

In [4]:

```
label_encoder = LabelEncoder()
df['sex'] = label_encoder.fit_transform(df['sex'])
df['embarked'] = label_encoder.fit_transform(df['embarked'].fillna('Unknown'))
```

In [5]:

```
# Fill in missing values with median values
df['age'] = df['age'].fillna(df['age'].median())
df['fare'] = df['fare'].fillna(df['fare'].median())
# Display the preprocessed DataFrame
print(df.head())
```

```
   passenger_id  pclass                                               nam
e  \
0          1216       3                             Smyth, Miss. Juli
a
1           699       3                                Cacic, Mr. Luk
a
2          1267       3  Van Impe, Mrs. Jean Baptiste (Rosalie Paula G
o...
3           449       2            Hocking, Mrs. Elizabeth (Eliza Need
s)
4           576       2                                Veal, Mr. Jame
s

   sex   age  sibsp  parch  ticket     fare cabin  embarked boat  body  \
0    0  28.0      0      0  335432   7.7333   NaN         1   13   NaN
1    1  38.0      0      0  315089   8.6625   NaN         2  NaN   NaN
2    0  30.0      1      1  345773  24.1500   NaN         2  NaN   NaN
3    0  54.0      1      3   29105  23.0000   NaN         2    4   NaN
4    1  40.0      0      0   28221  13.0000   NaN         2  NaN   NaN

                 home.dest  survived
0                      NaN         1
1                  Croatia         0
2                      NaN         0
3      Cornwall / Akron, OH         1
4  Barre, Co Washington, VT         0
```

In [6]:

```
scaler = StandardScaler()
numerical_cols = ['age', 'fare']
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

In [7]:

```
features = ['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']
target = 'survived'
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target])
# Display the shapes of the resulting data splits
print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)
print('X_test shape:', X_test.shape)
print('y_test shape:', y_test.shape)
```

```
X_train shape: (637, 7)
y_train shape: (637,)
X_test shape: (213, 7)
y_test shape: (213,)
```

In [8]:

```
mlp = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000, random_state=42)
mlp.fit(X_train, y_train)

# Print the accuracy of the classifier on the training data
train_accuracy = mlp.score(X_train, y_train)
print('Training accuracy:', train_accuracy)
```

```
Training accuracy: 0.8288854003139717
```

In [9]:

```
y_pred = mlp.predict(X_test)
# calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# print the results
print("Accuracy: {:.3f}".format(accuracy))
print("Precision: {:.3f}".format(precision))
print("Recall: {:.3f}".format(recall))
```

```
Accuracy: 0.808
Precision: 0.828
Recall: 0.639
```

In [ ]: