In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier

from deap import base, creator, tools, algorithms
import random
```

In [2]:
```
df = pd.read_csv('titanic.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [4]:
```
df.shape
```

Out[4]:
```
(891, 12)
```

In [5]:
```
df.describe()
```

Out[5]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [6]:
```
df.isna().sum()
```

Out[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [7]:
```
df = df[["Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked", "Survi
```

In [8]:
```
df.dropna(inplace=True)
```

In [9]:
```
label_enc = LabelEncoder()
df.loc[:, "Sex"] = label_enc.fit_transform(df["Sex"])
df.loc[:, "Embarked"] = label_enc.fit_transform(df["Embarked"])
```

In [10]:
```
X = df.drop("Survived", axis=1)
y = df["Survived"]
```

In [11]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2
```

In [13]:
```
# Set up DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
```

In [14]:
```
toolbox = base.Toolbox()
```

In [15]:
```
# Individual: [n_estimators, max_depth, min_samples_split]
toolbox.register("n_estimators", random.randint, 10, 200)
toolbox.register("max_depth", random.randint, 3, 20)
toolbox.register("min_samples_split", random.randint, 2, 10)

toolbox.register("individual", tools.initCycle, creator.Individual,
                 (toolbox.n_estimators, toolbox.max_depth, toolbox.min_samples
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

In [16]:
```
# Fitness function
def evaluate(ind):
    clf = RandomForestClassifier(
        n_estimators=ind[0],
        max_depth=ind[1],
        min_samples_split=ind[2],
        random_state=42
    )
    clf.fit(X_train, y_train)
    preds = clf.predict(X_test)
    return (accuracy_score(y_test, preds),)
```

In [17]:
```
toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxUniform, indpb=0.5)
toolbox.register("mutate", tools.mutUniformInt, low=[10, 3, 2], up=[200, 20, 1
toolbox.register("select", tools.selTournament, tournsize=3)
```

In [18]:
```
# Run GA
def run_deap():
    pop = toolbox.population(n=10)
    hof = tools.HallOfFame(1)
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("avg", np.mean)
    stats.register("max", np.max)

    pop, logbook = algorithms.eaSimple(
        pop, toolbox,
        cxpb=0.5, mutpb=0.2,
        ngen=10, stats=stats, halloffame=hof, verbose=True
    )

    print("\nBest parameters found:", hof[0])
    return hof[0]

best = run_deap()
```

```
gen     nevals  avg             max
0       10      0.786713        0.804196
1       3       0.798601        0.804196
2       9       0.801399        0.804196
3       4       0.804196        0.804196
4       8       0.804196        0.804196
5       7       0.803497        0.804196
6       1       0.804196        0.804196
7       6       0.804196        0.804196
8       2       0.804196        0.804196
9       4       0.804196        0.804196
10      4       0.803497        0.804196

Best parameters found: [109, 12, 9]
```

In [19]:
```
# Final model
final_model = RandomForestClassifier(
    n_estimators=best[0],
    max_depth=best[1],
    min_samples_split=best[2],
    random_state=42
)
final_model.fit(X_train, y_train)
final_preds = final_model.predict(X_test)

print("Final Test Accuracy:", accuracy_score(y_test, final_preds) * 100)
```

```
Final Test Accuracy: 80.41958041958041
```

In [ ]: