In [105]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
```

In [ ]:

```
https://data.world/marshalldatasolution/breast-cancer
```

In [106]:

```
df = pd.read_excel('https://query.data.world/s/xawnxtoyk3lc5rfkgmxydwu5vh7i
```

In [107]:

```
df
```

Out[107]:

	Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	Resistin	MCP.1	C
0	48	23.500000	70	2.707	0.467409	8.8071	9.702400	7.99585	417.114	
1	83	20.690495	92	3.115	0.706897	8.8438	5.429285	4.06405	468.786	
2	82	23.124670	91	4.498	1.009651	17.9393	22.432040	9.27715	554.697	
3	68	21.367521	77	3.226	0.612725	9.8827	7.169560	12.76600	928.220	
4	86	21.111111	92	3.549	0.805386	6.6994	4.819240	10.57635	773.920	
...	...	...	...	...	...	...	...	...	...	
111	45	26.850000	92	3.330	0.755688	54.6800	12.100000	10.96000	268.230	
112	62	26.840000	100	4.530	1.117400	12.4500	21.420000	7.32000	330.160	
113	65	32.050000	97	5.730	1.370998	61.4800	22.540000	10.33000	314.050	
114	72	25.590000	82	2.820	0.570392	24.9600	33.750000	3.27000	392.460	
115	86	27.180000	138	19.910	6.777364	90.2800	14.110000	4.35000	90.090	

116 rows × 10 columns

In [108]:

```
# Handle missing values if any
df.dropna(inplace=True)
```

In [109]:
```
# Step 2: Split data into features and target variable
X = df.drop(columns=["Classification"])
y = df["Classification"]
```

In [110]:
```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

Decision Tree Classifier

In [7]:
```
# Model training
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[7]:
```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

In [8]:
```
# Model evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.75

Classification Report:
               precision    recall  f1-score   support

           1       0.80      0.67      0.73        12
           2       0.71      0.83      0.77        12

    accuracy                           0.75        24
   macro avg       0.76      0.75      0.75        24
weighted avg       0.76      0.75      0.75        24
```

K Nearest Neighbour

In [22]:
```
#  Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [23]:
```
#  Model training
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train_scaled, y_train)
```

Out[23]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [24]:
```
# Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8333333333333334

Classification Report:
               precision    recall  f1-score   support

           1       0.79      0.92      0.85        12
           2       0.90      0.75      0.82        12

    accuracy                           0.83        24
   macro avg       0.84      0.83      0.83        24
weighted avg       0.84      0.83      0.83        24
```

Logistic Regression

In [32]:
```
# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [33]:
```
#  Model training
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

Out[33]:
```
▾ LogisticRegression

LogisticRegression()
```

In [34]:
```
# Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.875

Classification Report:
               precision    recall  f1-score   support

           1       0.85      0.92      0.88        12
           2       0.91      0.83      0.87        12

    accuracy                           0.88        24
   macro avg       0.88      0.88      0.87        24
weighted avg       0.88      0.88      0.87        24
```

Naive Bayes

In [41]:
```
# Model training
model = GaussianNB()
model.fit(X_train, y_train)
```

Out[41]:
```
▾ GaussianNB
GaussianNB()
```

In [42]:
```
#  Model evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.75

Classification Report:
               precision    recall  f1-score   support

           1       0.69      0.92      0.79        12
           2       0.88      0.58      0.70        12

    accuracy                           0.75        24
   macro avg       0.78      0.75      0.74        24
weighted avg       0.78      0.75      0.74        24
```

Ada Boosting

In [51]:
```
#  Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [52]:
```
#  Model training
model = AdaBoostClassifier(n_estimators=50, random_state=42)
model.fit(X_train_scaled, y_train)
```

Out[52]:
```
▾         AdaBoostClassifier
AdaBoostClassifier(random_state=42)
```

In [53]:
```
# Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.75

Classification Report:
               precision    recall  f1-score   support

           1       0.69      0.92      0.79        12
           2       0.88      0.58      0.70        12

    accuracy                           0.75        24
   macro avg       0.78      0.75      0.74        24
weighted avg       0.78      0.75      0.74        24
```

Extra Tree

In [60]:
```
#  Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [61]:
```
#  Model training
model  = ExtraTreesClassifier(n_estimators=100, random_state=42)  # You can
model.fit(X_train_scaled, y_train)
```

Out[61]:
```
▾         ExtraTreesClassifier
ExtraTreesClassifier(random_state=42)
```

In [62]:

```
# Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.875

Classification Report:
               precision    recall  f1-score   support

           1       0.80      1.00      0.89        12
           2       1.00      0.75      0.86        12

    accuracy                           0.88        24
   macro avg       0.90      0.88      0.87        24
weighted avg       0.90      0.88      0.87        24
```

Gradient Boosting

In [68]:

```
#  Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [69]:

```
#  Model training
model = GradientBoostingClassifier(n_estimators=100, random_state=42)  # Yo
model.fit(X_train_scaled, y_train)
```

Out[69]:

```
▾         GradientBoostingClassifier
GradientBoostingClassifier(random_state=42)
```

In [70]:

```
#  Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.875

Classification Report:
               precision    recall  f1-score   support

           1       1.00      0.75      0.86        12
           2       0.80      1.00      0.89        12

    accuracy                           0.88        24
   macro avg       0.90      0.88      0.87        24
weighted avg       0.90      0.88      0.87        24
```

Multi layer perceptron

In [76]:
```
#  Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [77]:
```
# Model training
model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000, random_s
model.fit(X_train_scaled, y_train)
```

Out[77]:
```
▾                                   MLPClassifier

MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=
42)
```

In [78]:
```
#  Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9166666666666666

Classification Report:
               precision    recall  f1-score   support

           1       0.92      0.92      0.92        12
           2       0.92      0.92      0.92        12

    accuracy                           0.92        24
   macro avg       0.92      0.92      0.92        24
weighted avg       0.92      0.92      0.92        24
```

Support Vector Machine

In [84]:
```
#  Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [85]:
```
#  Model training
model = SVC(kernel='rbf', random_state=42)
model.fit(X_train_scaled, y_train)
```

Out[85]:
```
▾         SVC

SVC(random_state=42)
```

In [86]:

```
#  Model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.875

Classification Report:
               precision    recall  f1-score   support

           1       0.85      0.92      0.88        12
           2       0.91      0.83      0.87        12

    accuracy                           0.88        24
   macro avg       0.88      0.88      0.87        24
weighted avg       0.88      0.88      0.87        24
```

Chat GPT

In [93]:

```
# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [94]:

```
# Train the model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

Out[94]:

```
▾ LogisticRegression

LogisticRegression()
```

In [95]:

```
# Make predictions
y_pred = model.predict(X_test_scaled)
```

In [96]:

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.875
```

In [104]:

```
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```

Classification Report:
               precision    recall  f1-score   support

           1       0.85      0.92      0.88        12
           2       0.91      0.83      0.87        12

    accuracy                           0.88        24
   macro avg       0.88      0.88      0.87        24
weighted avg       0.88      0.88      0.87        24

```

Random forest

In [111]:

```
# Train the Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[111]:

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

In [112]:

```
# Make predictions
y_pred = model.predict(X_test)
```

In [113]:

```

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7916666666666666
```

In [114]:

```
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```

Classification Report:
               precision    recall  f1-score   support

           1       0.82      0.75      0.78        12
           2       0.77      0.83      0.80        12

    accuracy                           0.79        24
   macro avg       0.79      0.79      0.79        24
weighted avg       0.79      0.79      0.79        24

```

In [ ]:

```

```