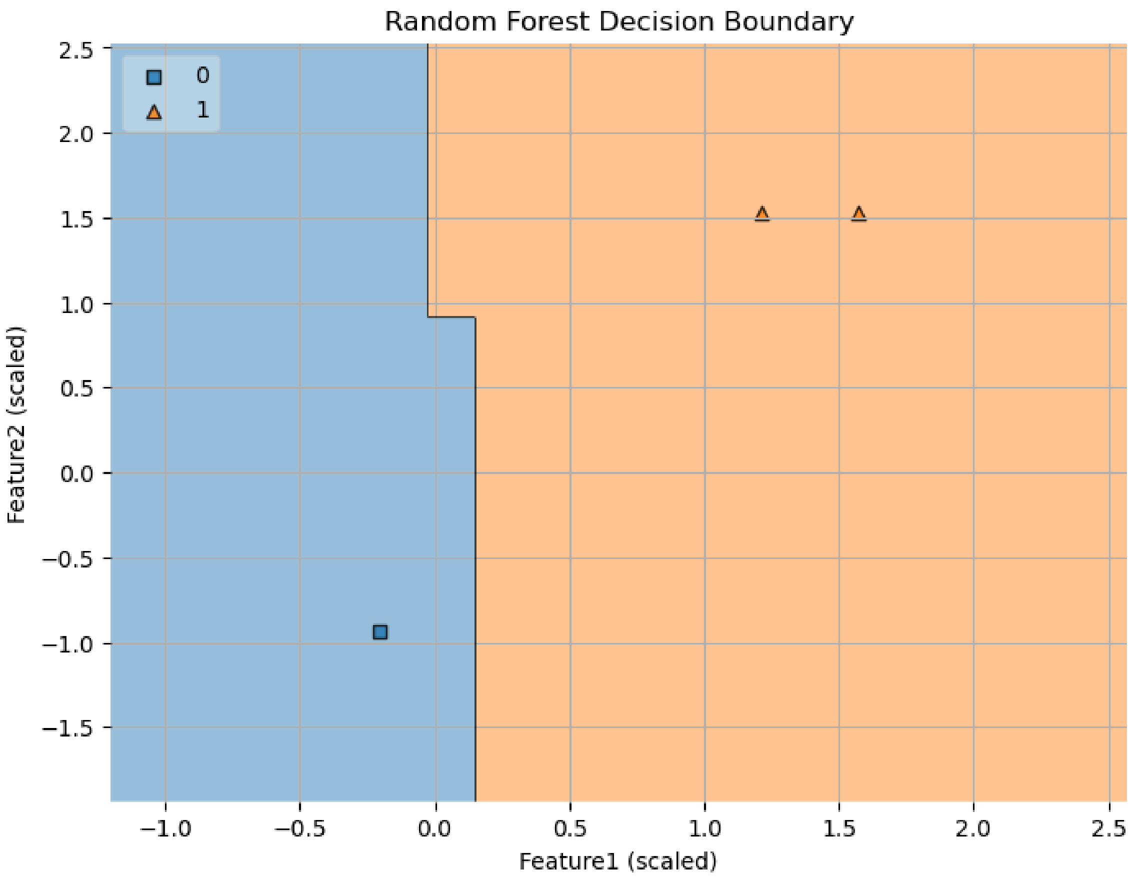
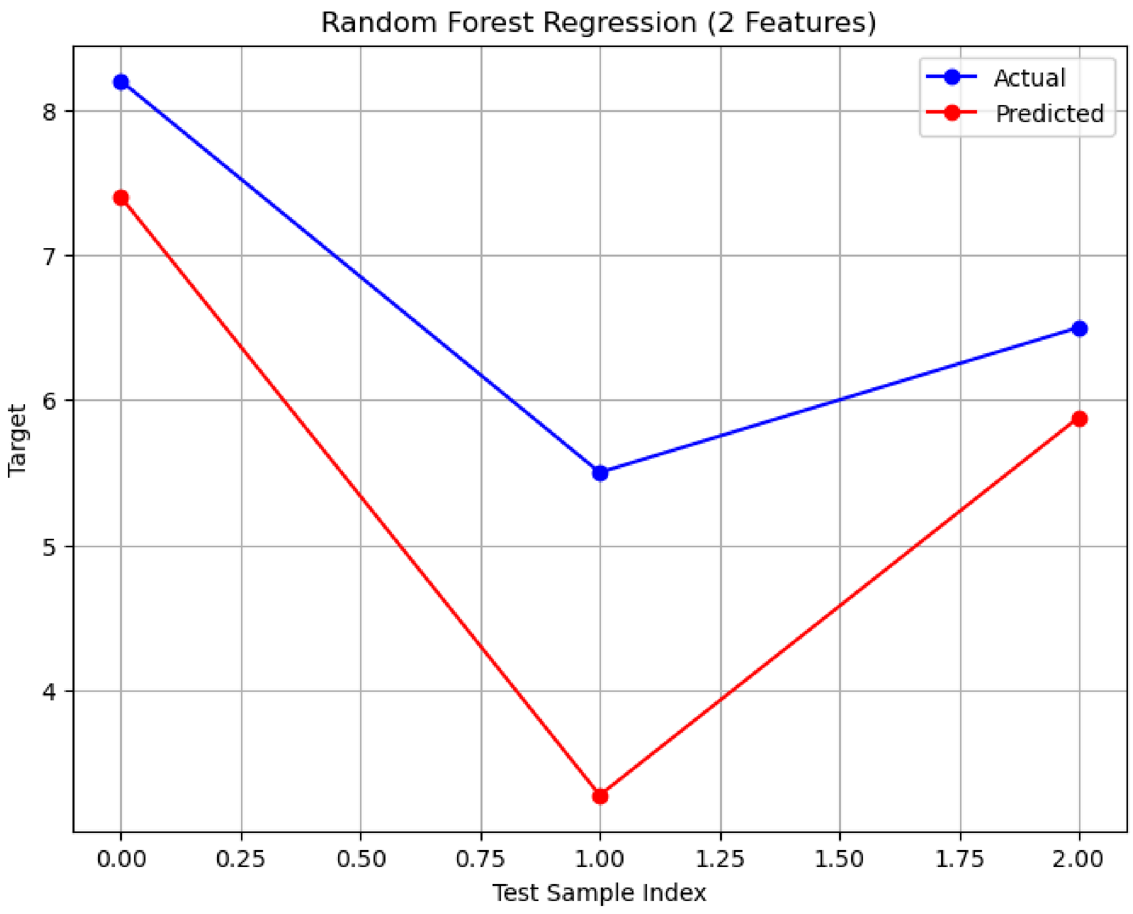
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from mlxtend.plotting import plot_decision_regions

data = {
    'Feature1': [2, 4, 4, 6, 6, 8, 1, 3, 9, 10],
    'Feature2': [4, 2, 4, 6, 4, 8, 2, 1, 8, 9],
    'Label': [0, 0, 0, 1, 1, 1, 0, 0, 1, 1]
}

df = pd.DataFrame(data)

csv_file = "random_forest_sample_data.csv"
df.to_csv(csv_file, index=False)

df = pd.read_csv(csv_file)

X = df[['Feature1', 'Feature2']].values
y = df['Label'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar

scaler = StandardScaler()
X_train_scaled = scaler.fit_t vransform(X_train)
X_test_scaled = scaler.transform(X_test)

rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train_scaled, y_train)

y_pred = rf_clf.predict(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

plt.figure(figsize=(8, 6))
plot_decision_regions(X_test_scaled, y_test, clf=rf_clf, legend=2)
plt.title("Random Forest Decision Boundary")
plt.xlabel("Feature1 (scaled)")
plt.ylabel("Feature2 (scaled)")
plt.grid(True)
plt.show()
```

```
Accuracy: 1.0
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```


Random Forest Decision Boundary
0
1
Feature2 (scaled)
Feature1 (scaled)
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler


data = {
    'Feature1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Feature2': [2, 1, 3, 4, 2, 5, 6, 5, 7, 8],
    'Target': [2.0, 2.5, 3.0, 4.0, 5.5, 6.5, 7.0, 8.2, 9.0, 10.0]
}

df = pd.DataFrame(data)


csv_file = "two_feature_regression_data.csv"
df.to_csv(csv_file, index=False)
df = pd.read_csv(csv_file)


X = df[['Feature1', 'Feature2']].values
y = df['Target'].values


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shu


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train_scaled, y_train)


y_pred = rf.predict(X_test_scaled)
print("R^2 Score:", r2_score(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))

plt.figure(figsize=(8, 6))
x_axis = range(len(y_test))
plt.plot(x_axis, y_test, 'bo-', label="Actual")
plt.plot(x_axis, y_pred, 'ro-', label="Predicted")
plt.title("Random Forest Regression (2 Features)")
plt.xlabel("Test Sample Index")
plt.ylabel("Target")
plt.legend()
plt.grid(True)
plt.show()
```

```
R^2 Score: -0.6033161896243298
Mean Squared Error: 1.9916749999999999
```

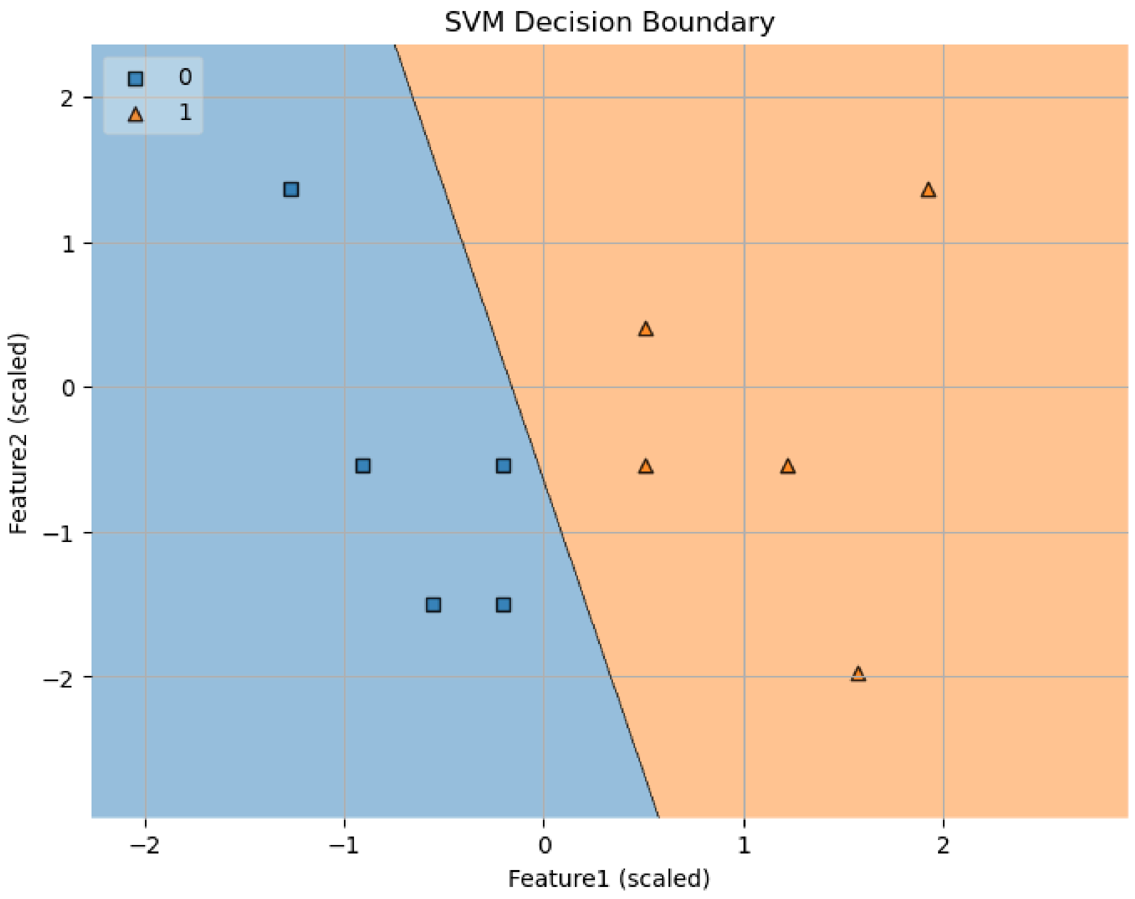

Random Forest Regression (2 Features)
Actual
Predicted
Target
Test Sample Index
8
7
6
5
4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

In [3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

data = {
    'Feature1':[2.5,1.3,3.3,2.1,1.1,3.0,1.5,2.8,1.0,2.7],
    'Feature2':[1.7,3.5,2.1,1.9,3.0,2.2,3.2,2.0,3.1,1.8],
    'Class':['A','B','A','A','B','A','B','A','B','A']
}

df = pd.DataFrame(data)

csv_filename = "Sample_data.csv"

df.to_csv(csv_filename, index = False)
print(f"sample data saved to {csv_filename}")

X = df[['Feature1','Feature2']]
y = df['Class']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_st

model = GaussianNB()
model.fit(X_train,y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test,y_pred)
report = classification_report(y_test,y_pred)

print("\n---Evaluation---")
print(f"Accuarcy: {accuracy*100:.2f}%")
print("\nClassification Report:\n",report)
```

```
sample data saved to Sample_data.csv

---Evaluation---
Accuarcy: 100.00%

Classification Report:
               precision    recall  f1-score   support

           A       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from mlxtend.plotting import plot_decision_regions
data = {
 'Feature1': [2, 4, 4, 6, 6, 8, 1, 3, 9, 10],
 'Feature2': [4, 2, 4, 4, 6, 4, 8, 2, 1, 8],
 'Label': [0, 0, 0, 1, 1, 1, 0, 0, 1, 1]
}
df = pd.DataFrame(data)
csv_file = "svm_sample_data.csv"
df.to_csv(csv_file, index=False)
df = pd.read_csv(csv_file)
X = df[['Feature1', 'Feature2']].values
y = df['Label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_scaled = scaler.transform(X)
svm_clf = SVC(kernel='linear')
svm_clf.fit(X_train_scaled, y_train)
y_pred = svm_clf.predict(X_test_scaled)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
plt.figure(figsize=(8, 6))
plot_decision_regions(X_scaled, y, clf=svm_clf, legend=2)
plt.title("SVM Decision Boundary")
plt.xlabel("Feature1 (scaled)")
plt.ylabel("Feature2 (scaled)")
plt.grid(True)
plt.show()
```

```
Accuracy: 1.0
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```


SVM Decision Boundary
■ 0
▲ 1
2
1
0
−1
−2
−2
−1
0
1
2
Feature1 (scaled)
Feature2 (scaled)

```
In [ ]:
```