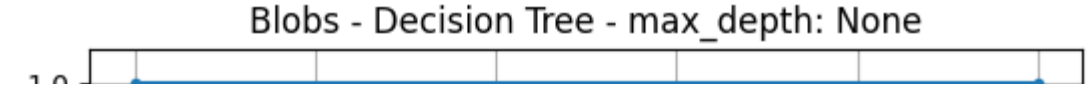
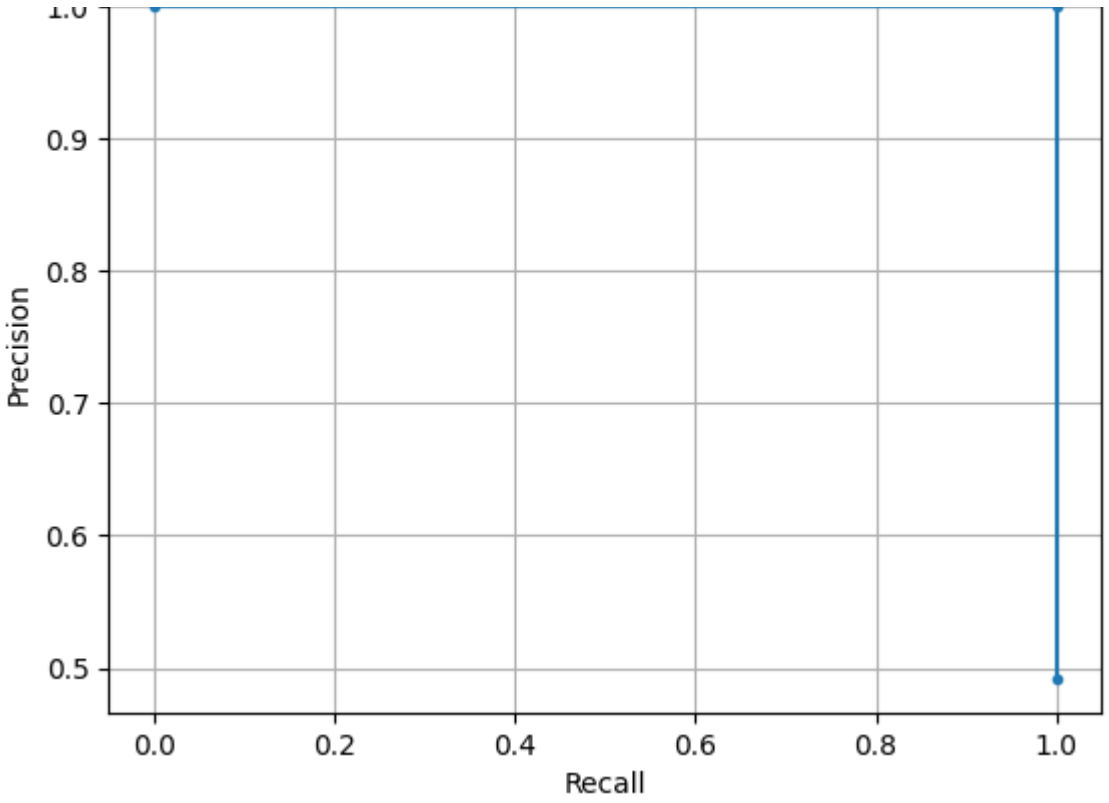
```
import numpy as np
from sklearn.datasets import load_iris, load_digits, load_breast_cancer, load_wi
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Gradien
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
import matplotlib.pyplot as plt


# Generate synthetic datasets
blobs_X, blobs_y = make_blobs(n_samples=1000, centers=2, random_state=42)
circles_X, circles_y = make_circles(n_samples=1000, noise=0.1, factor=0.5, rando
moons_X, moons_y = make_moons(n_samples=1000, noise=0.1, random_state=42)


# Define datasets
datasets = {
    "Blobs": (blobs_X, blobs_y),
    "Circles": (circles_X, circles_y),
    "Moons": (moons_X, moons_y)
}



# Define parameter configurations for each classifier
model_params = {
    "Decision Tree": {"max_depth": [None, 3], "min_samples_split": [2, 5]},
    "Random Forest": {"n_estimators": [100, 200], "max_depth": [None, 5]},
    "AdaBoost": {"n_estimators": [50, 100], "learning_rate": [1.0, 0.5]},
    "Gradient Boost": {"n_estimators": [50, 100], "learning_rate": [0.1, 0.05]}
}



# Initialize a dictionary to store the results
results = {}

# Loop through each dataset
for dataset_name, (X, y) in datasets.items():
    print(f"Dataset: {dataset_name}")

    results[dataset_name] = {}

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

    # Loop through each classifier and its parameter configurations
    for model_name, params in model_params.items():
        print(f"Training {model_name}...")
        results[dataset_name][model_name] = {}

        for param_name, param_values in params.items():
            for param_value in param_values:
                # Create and train the classifier with the current parameter con
```

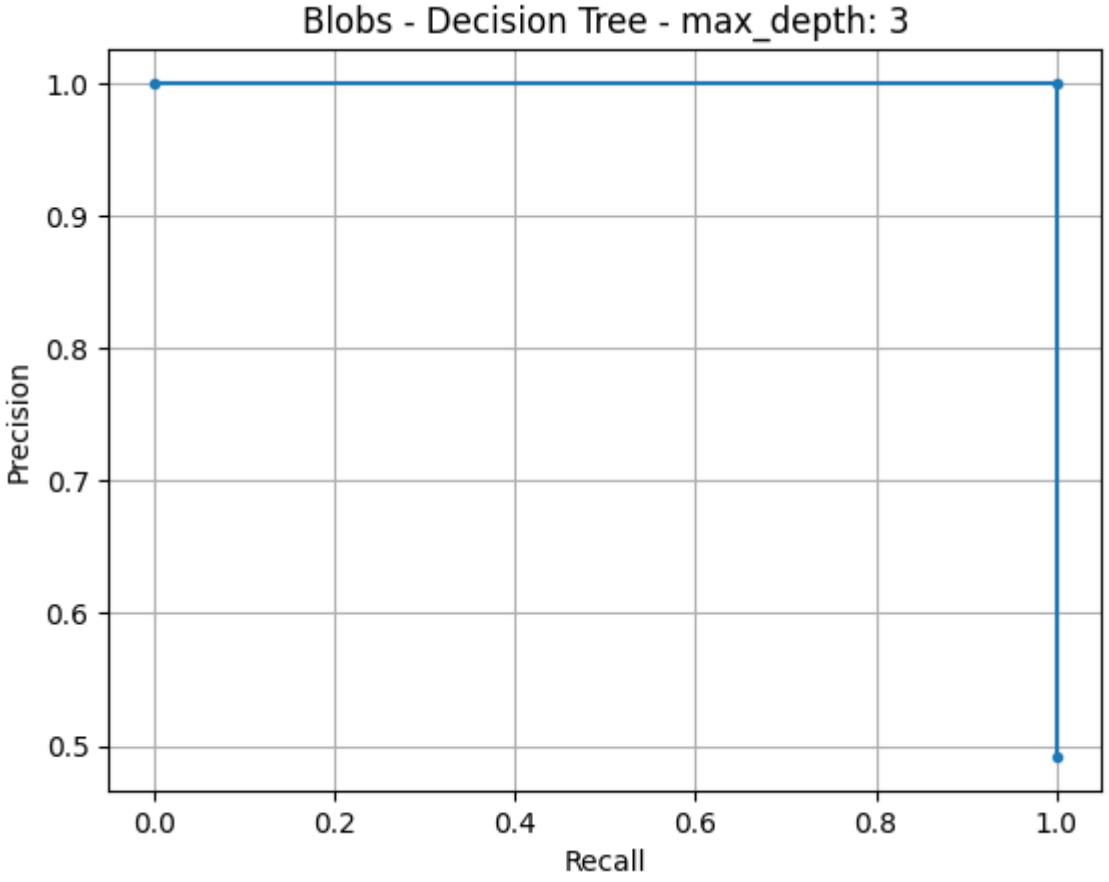
```
            if model_name == "Decision Tree":
                classifier = DecisionTreeClassifier(**{param_name: param_valu
            elif model_name == "Random Forest":
                classifier = RandomForestClassifier(**{param_name: param_valu
            elif model_name == "AdaBoost":
                classifier = AdaBoostClassifier(**{param_name: param_value})
            elif model_name == "Gradient Boost":
                classifier = GradientBoostingClassifier(**{param_name: param_
            else:
                raise ValueError(f"Unknown model name: {model_name}")

            classifier.fit(X_train, y_train)

            # Make predictions on the test set
            y_pred = classifier.predict(X_test)

            # Calculate evaluation metrics
            accuracy = accuracy_score(y_test, y_pred)
            precision = precision_score(y_test, y_pred, average="weighted")
            recall = recall_score(y_test, y_pred, average="weighted")
            f1 = f1_score(y_test, y_pred, average="weighted")

            # Calculate precision-recall curve
            precision_curve, recall_curve, _ = precision_recall_curve(y_test
            area_under_curve = auc(recall_curve, precision_curve)

            # Store the results
            results[dataset_name][model_name][(param_name, param_value)] = {
                "Accuracy": accuracy,
                "Precision": precision,
                "Recall": recall,
                "F1-score": f1,
                "Precision-Recall AUC": area_under_curve
            }
            print(f"{param_name}: {param_value}, Accuracy: {accuracy:.2f}, P:

            # Plot precision-recall curve and save figure
            plt.figure()
            plt.plot(recall_curve, precision_curve, marker='.')
            plt.title(f'{dataset_name} - {model_name} - {param_name}: {param_
            plt.xlabel('Recall')
            plt.ylabel('Precision')
            plt.grid(True)
            plt.savefig(f'{dataset_name}_{model_name}_{param_name}_{param_va
```

```
Dataset: Blobs
Training Decision Tree...
max_depth: None, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1
max_depth: 3, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1.00
min_samples_split: 2, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-sco
min_samples_split: 5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-sco
Training Random Forest...
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 200, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
```

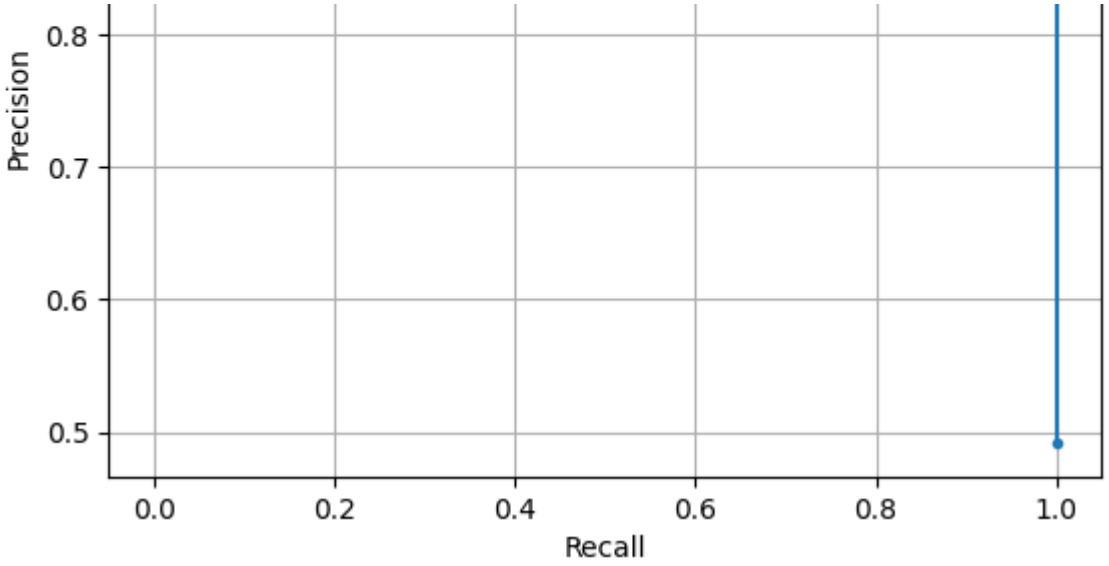
```
max_depth: None, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1
max_depth: 5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1.00
Training AdaBoost...
n_estimators: 50, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
learning_rate: 1.0, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
learning_rate: 0.5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
Training Gradient Boost...
n_estimators: 50, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
learning_rate: 0.1, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
learning_rate: 0.05, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-scor
Dataset: Circles
Training Decision Tree...
max_depth: None, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score: 0
max_depth: 3, Accuracy: 0.82, Precision: 0.84, Recall: 0.82, F1-score: 0.82
min_samples_split: 2, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-sco
min_samples_split: 5, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-sco
Training Random Forest...
n_estimators: 100, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score:
<ipython-input-5-41a8113a7968>:58: RuntimeWarning: More than 20 figures hav
  plt.figure()
n_estimators: 200, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score:
max_depth: None, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score: 0
max_depth: 5, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score: 0.96
Training AdaBoost...
n_estimators: 50, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score:
n_estimators: 100, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score:
learning_rate: 1.0, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score
learning_rate: 0.5, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score
Training Gradient Boost...
n_estimators: 50, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score:
n_estimators: 100, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score:
learning_rate: 0.1, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score
learning_rate: 0.05, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-scor
Dataset: Moons
Training Decision Tree...
max_depth: None, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1
max_depth: 3, Accuracy: 0.93, Precision: 0.94, Recall: 0.93, F1-score: 0.93
min_samples_split: 2, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-sco
min_samples_split: 5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-sco
Training Random Forest...
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 200, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
max_depth: None, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1
max_depth: 5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1.00
Training AdaBoost...
n_estimators: 50, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
learning_rate: 1.0, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
learning_rate: 0.5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
Training Gradient Boost...
n_estimators: 50, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
learning_rate: 0.1, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
learning_rate: 0.05, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-scor
```

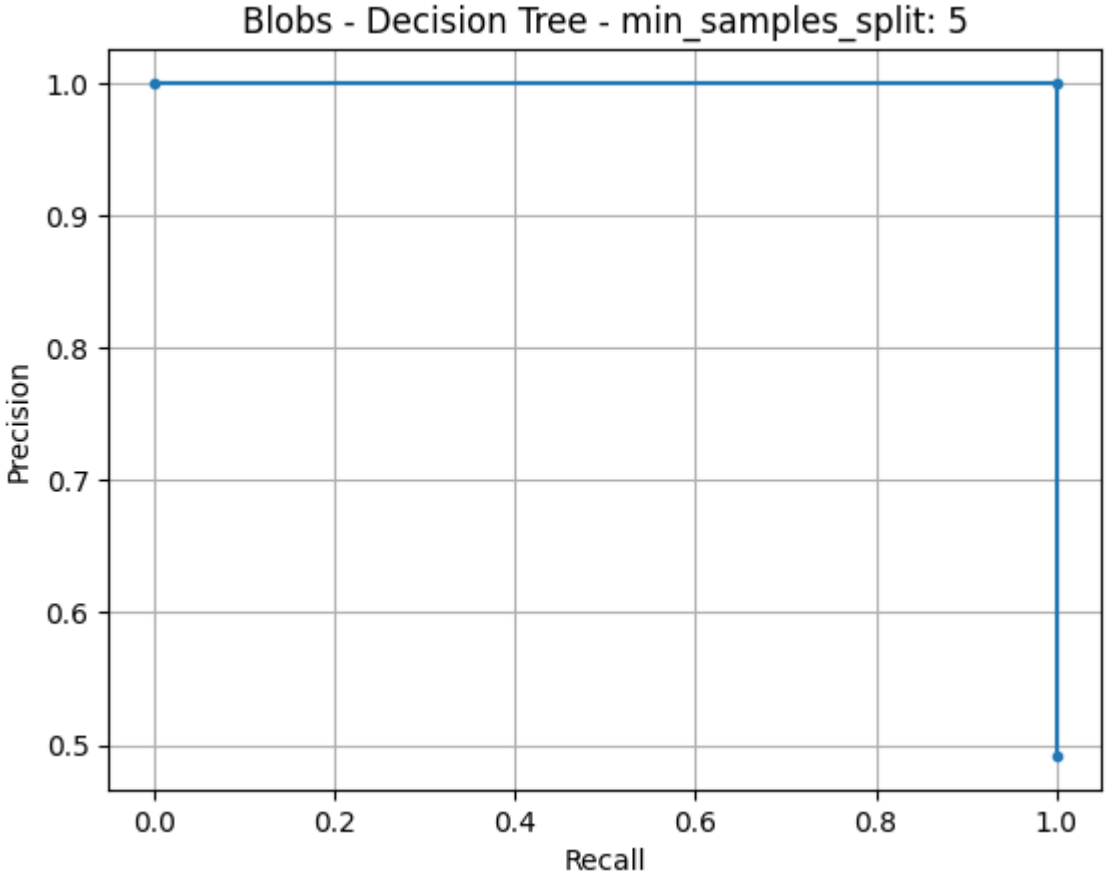


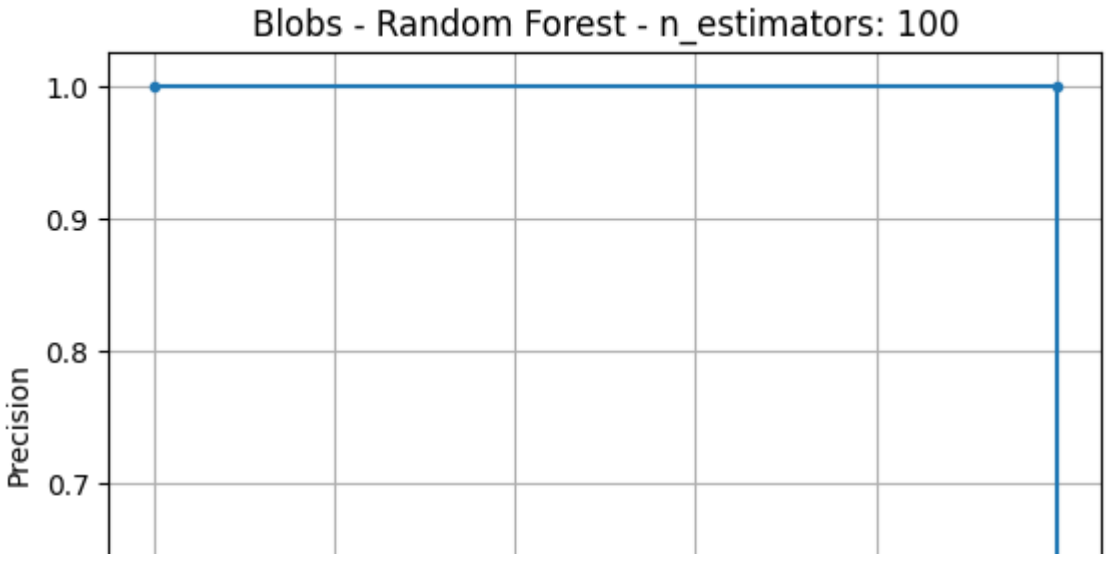

1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

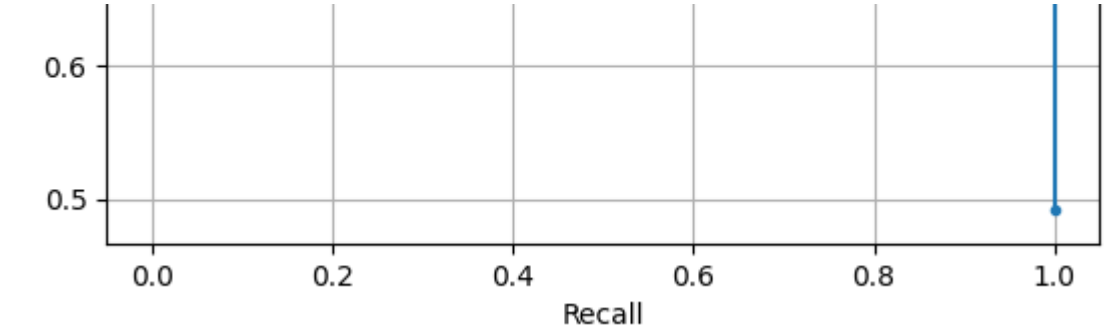

Blobs - Decision Tree - max_depth: 3
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

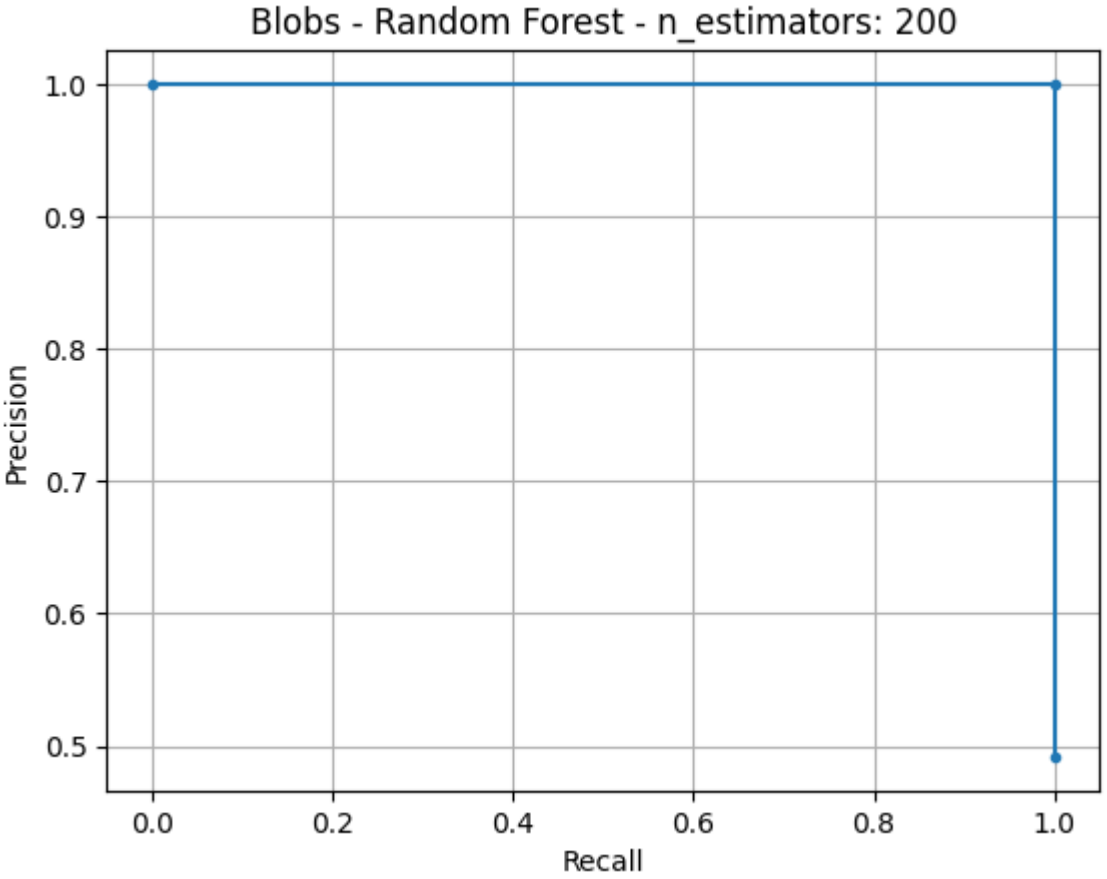

Blobs - Decision Tree - min_samples_split: 2
1.0
0.9

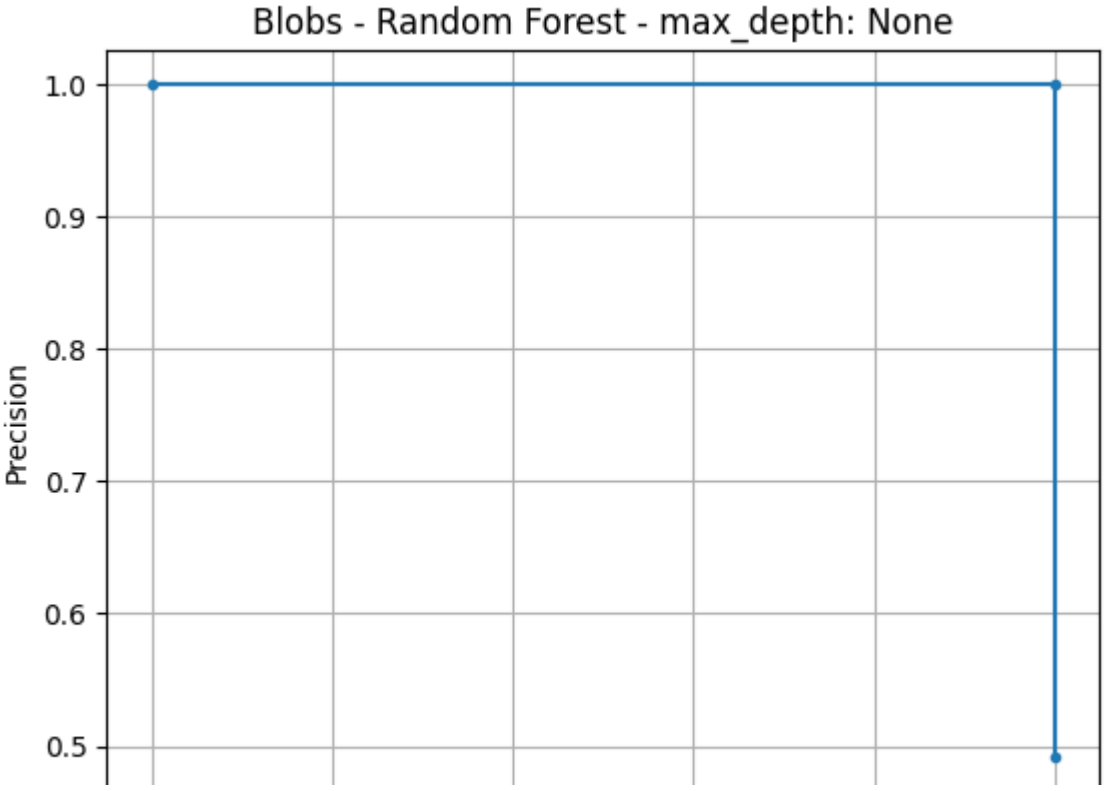

Precision
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

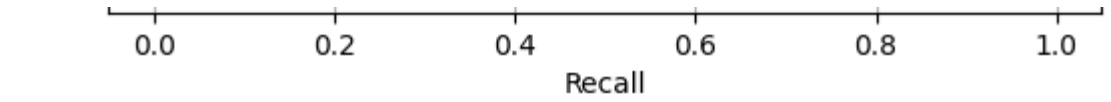

Blobs - Decision Tree - min_samples_split: 5
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

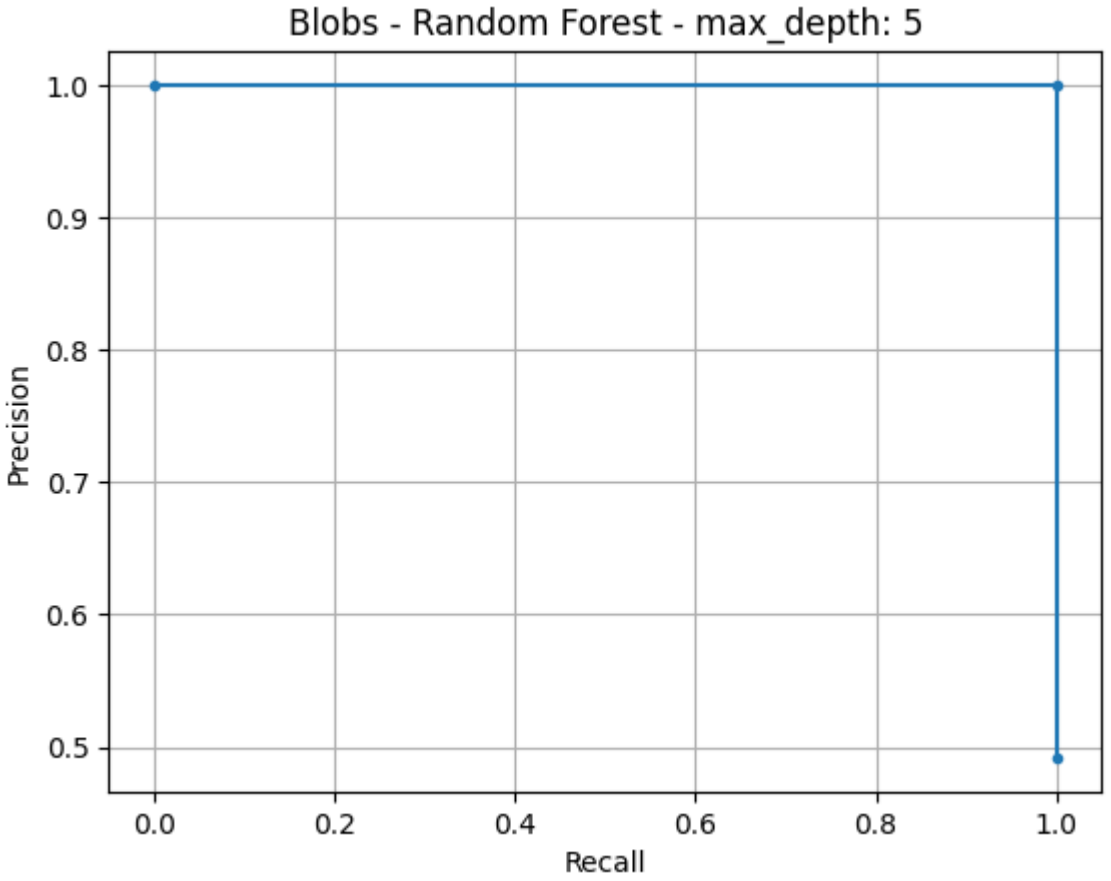

Blobs - Random Forest - n_estimators: 100
Precision
1.0
0.9
0.8
0.7

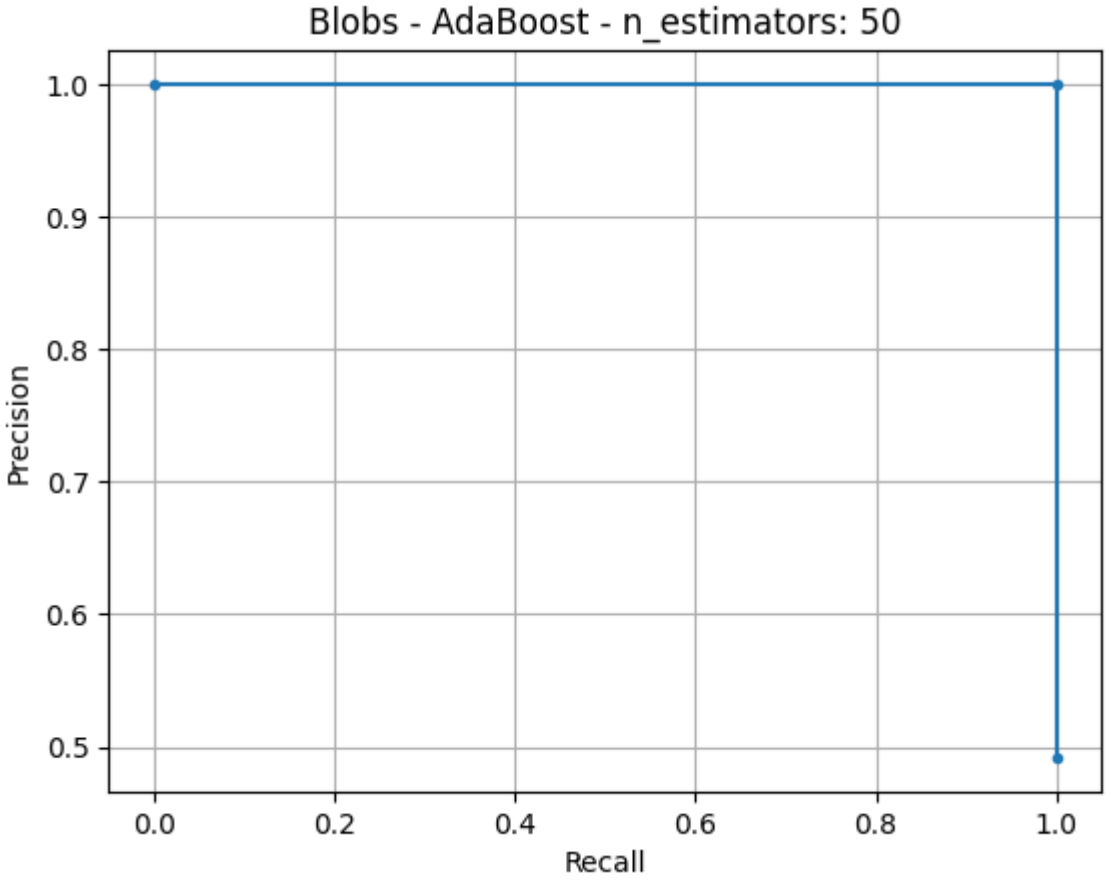

0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

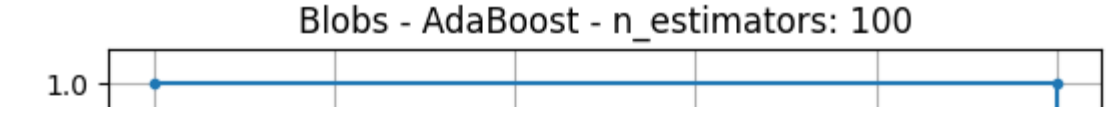

Blobs - Random Forest - n_estimators: 200
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

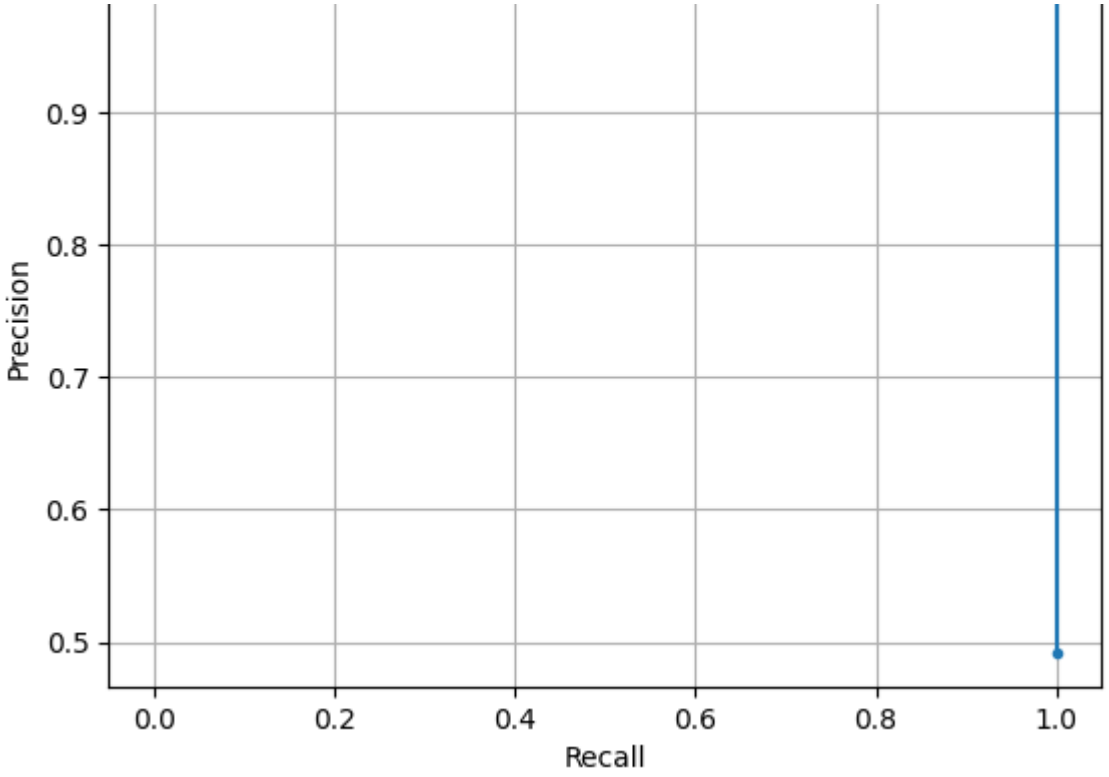

Blobs - Random Forest - max_depth: None
1.0
0.9
0.8
0.7
0.6
0.5
Precision

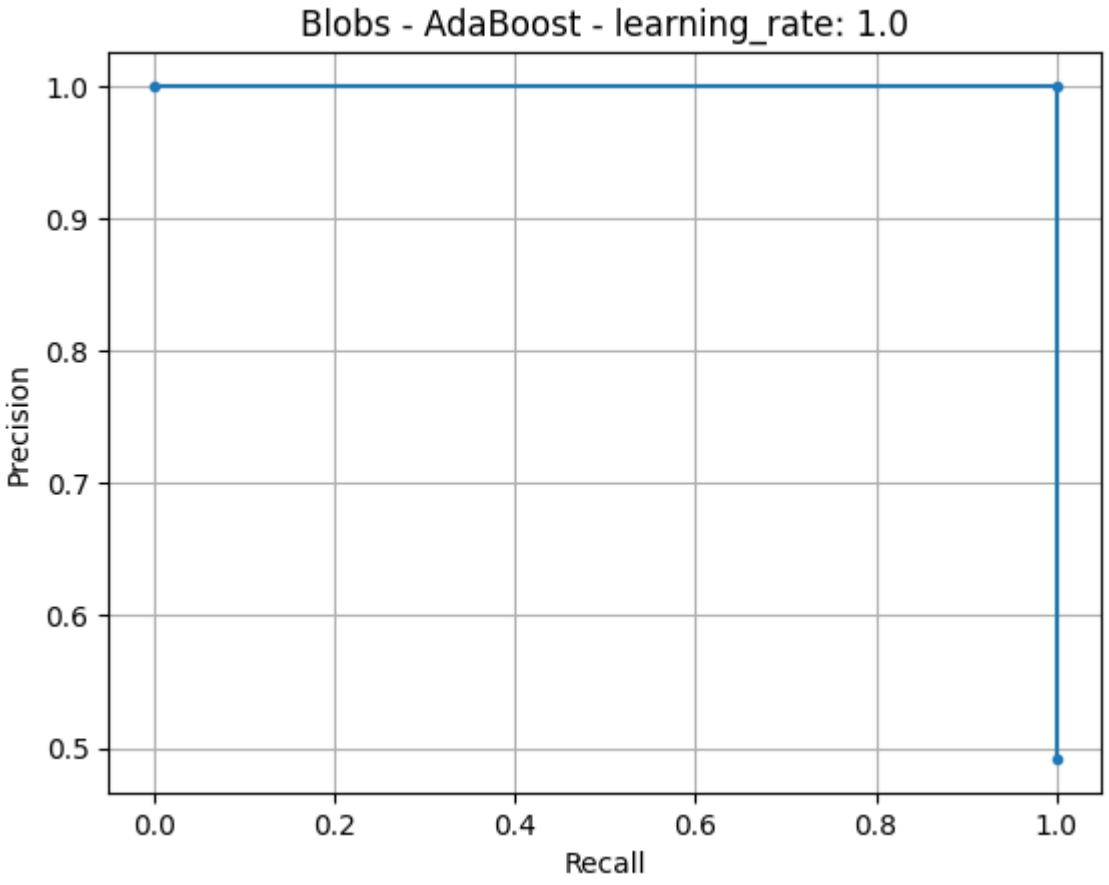

0.0
0.2
0.4
0.6
0.8
1.0
Recall

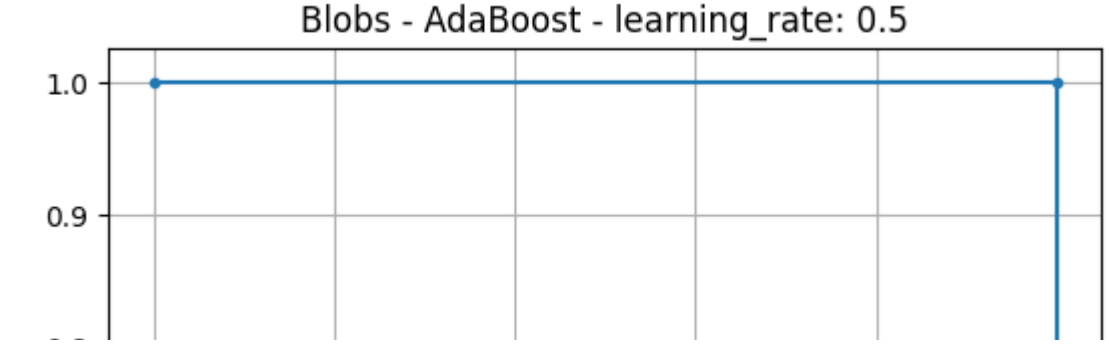

Blobs - Random Forest - max_depth: 5
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

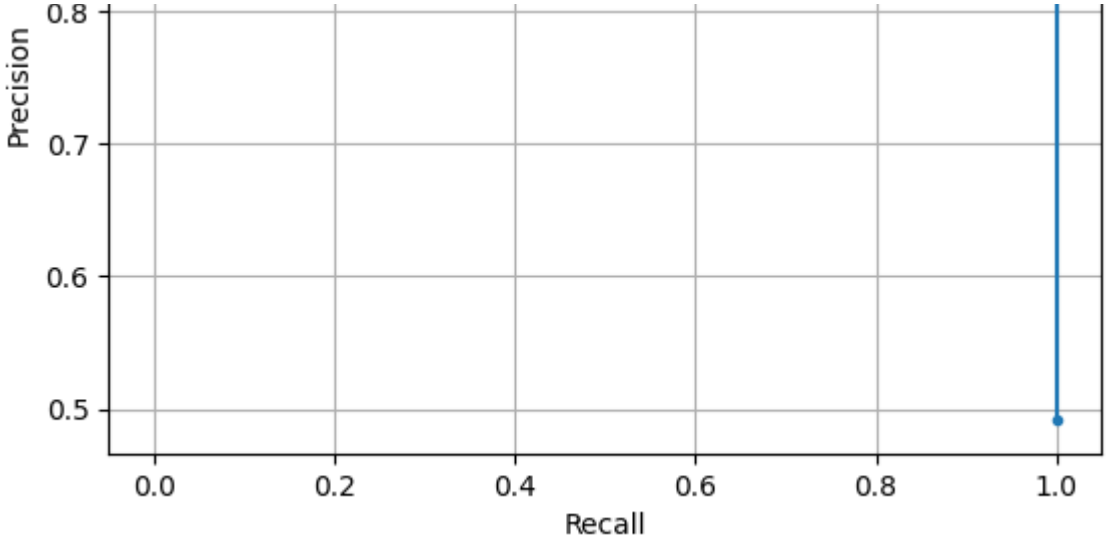

Blobs - AdaBoost - n_estimators: 50
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

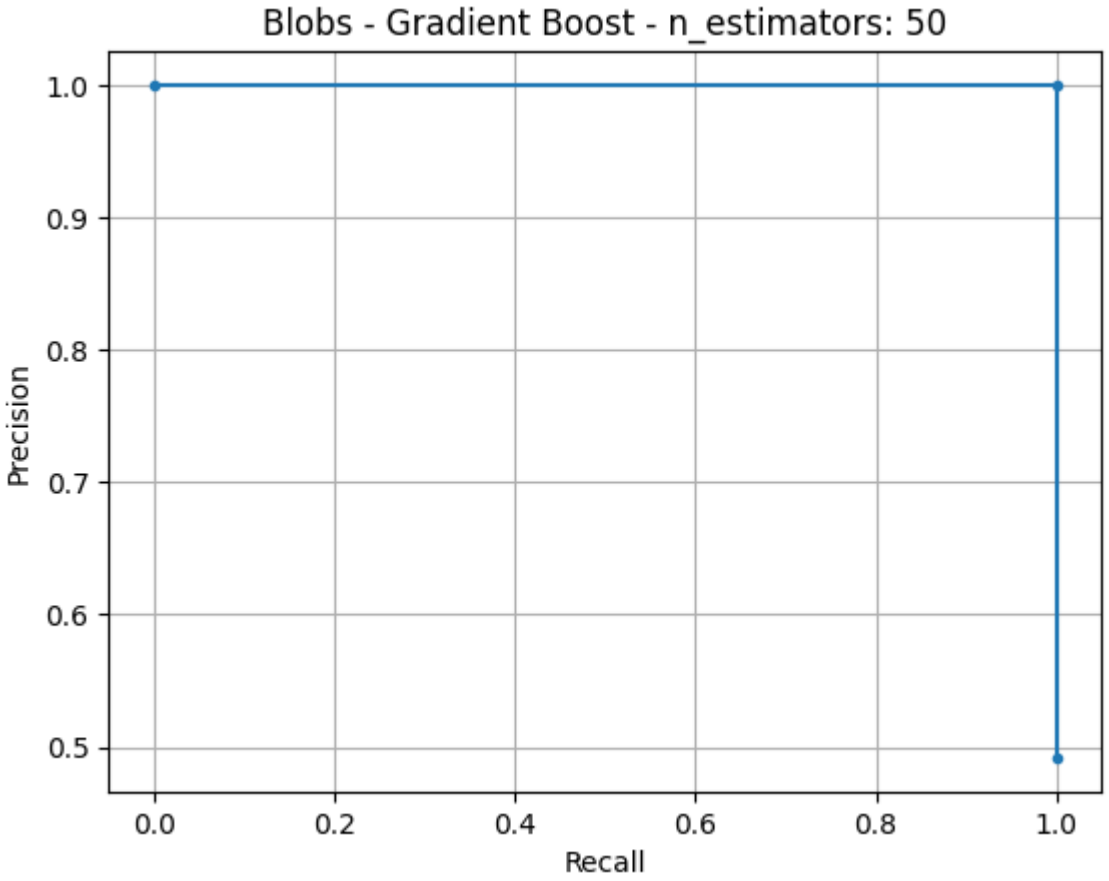

Blobs - AdaBoost - n_estimators: 100
1.0

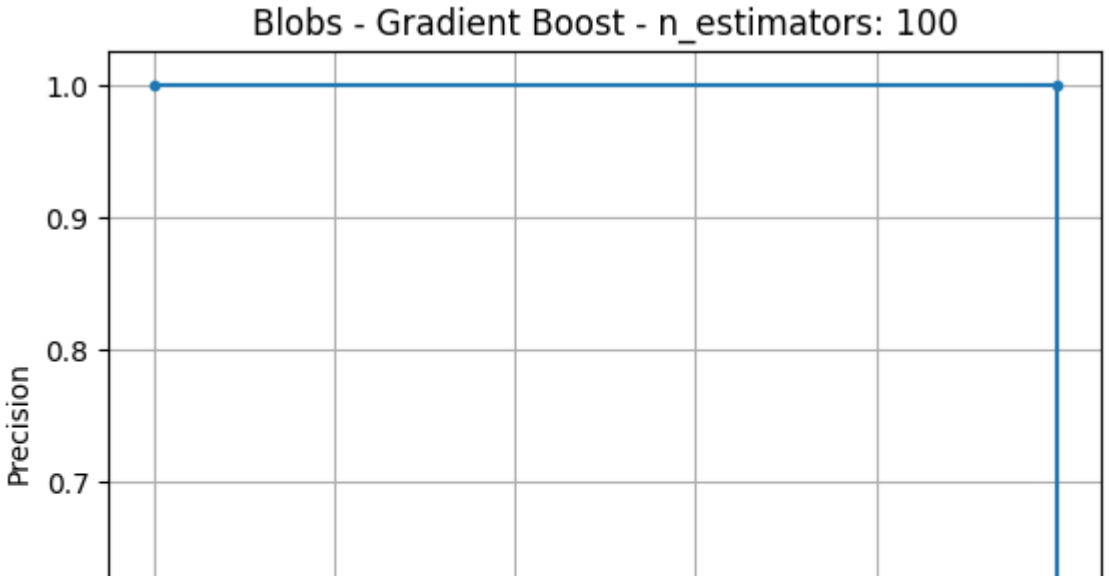

0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

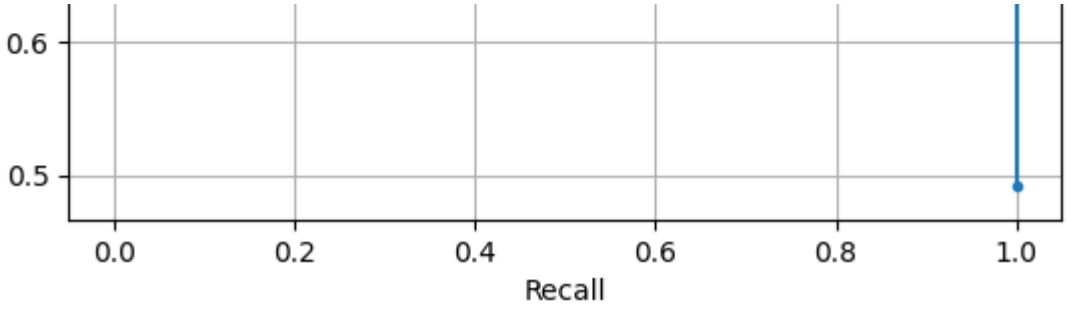

Blobs - AdaBoost - learning_rate: 1.0
1.0
0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

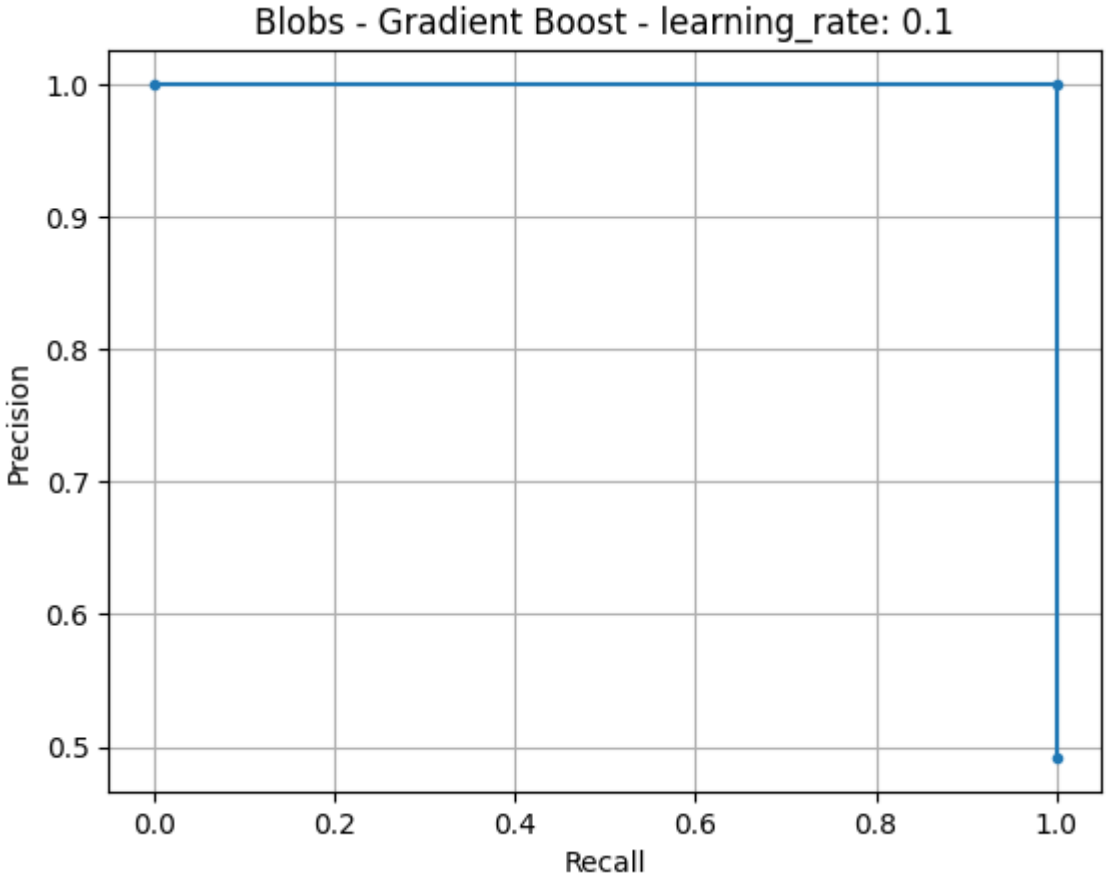

Blobs - AdaBoost - learning_rate: 0.5
1.0
0.9

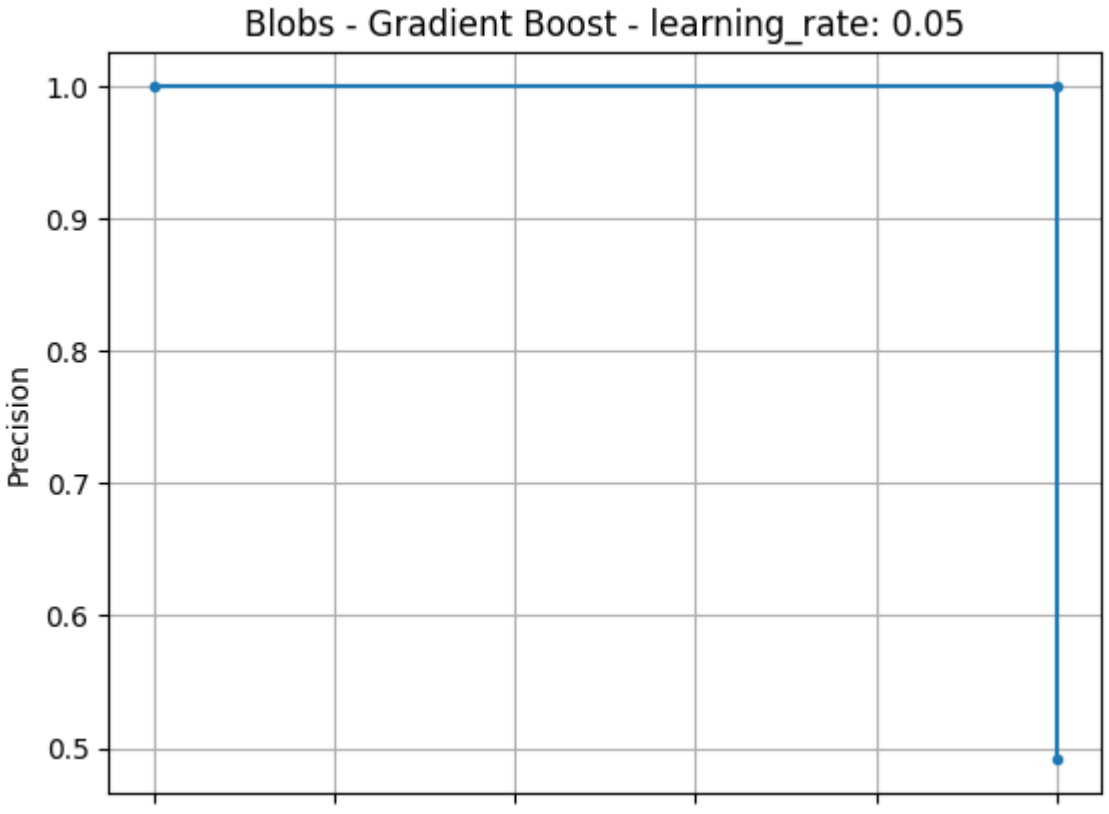

Precision
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

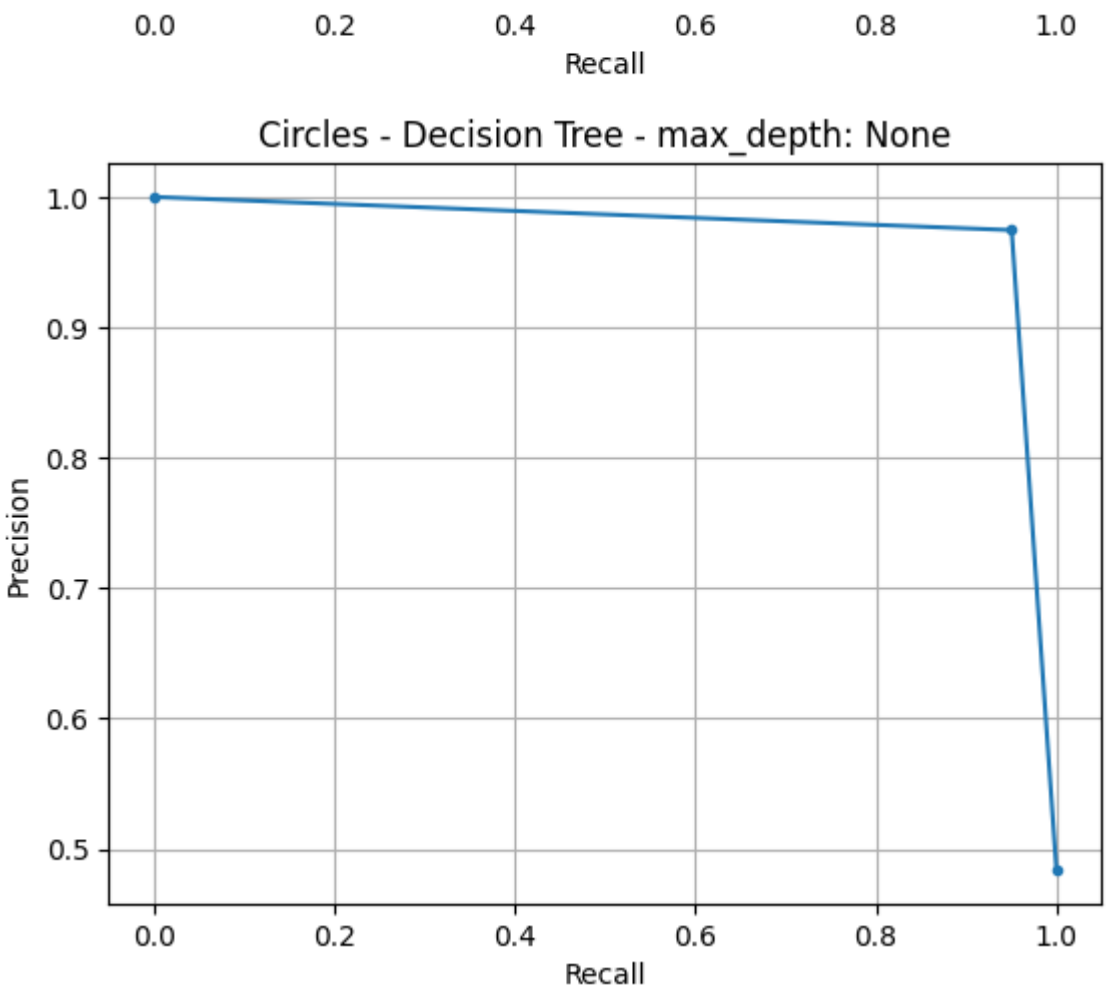

Blobs - Gradient Boost - n_estimators: 50
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

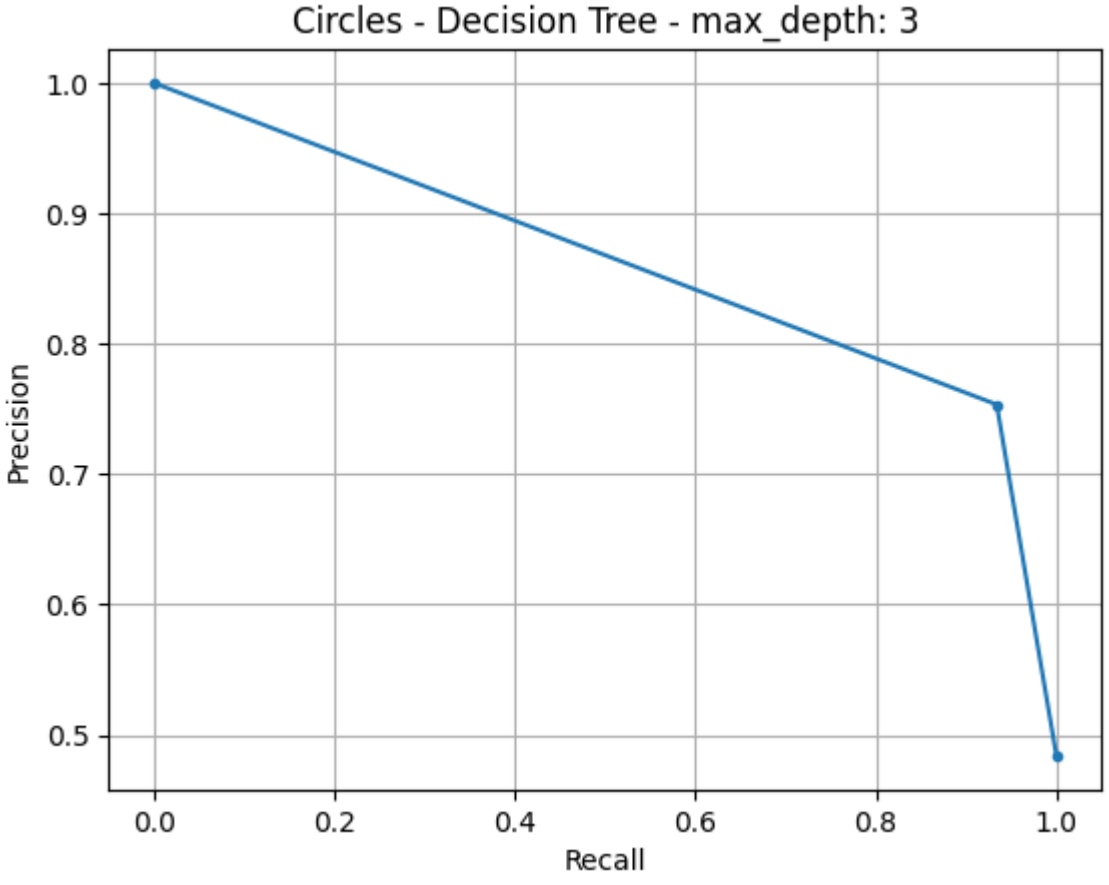

Blobs - Gradient Boost - n_estimators: 100
Precision
1.0
0.9
0.8
0.7

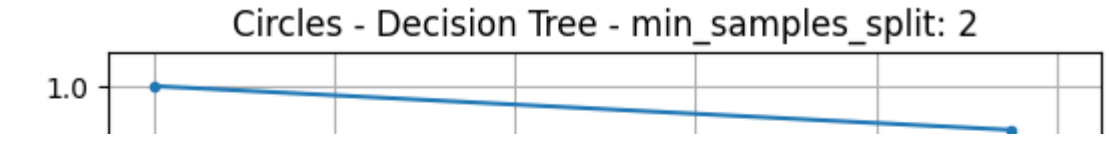

0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

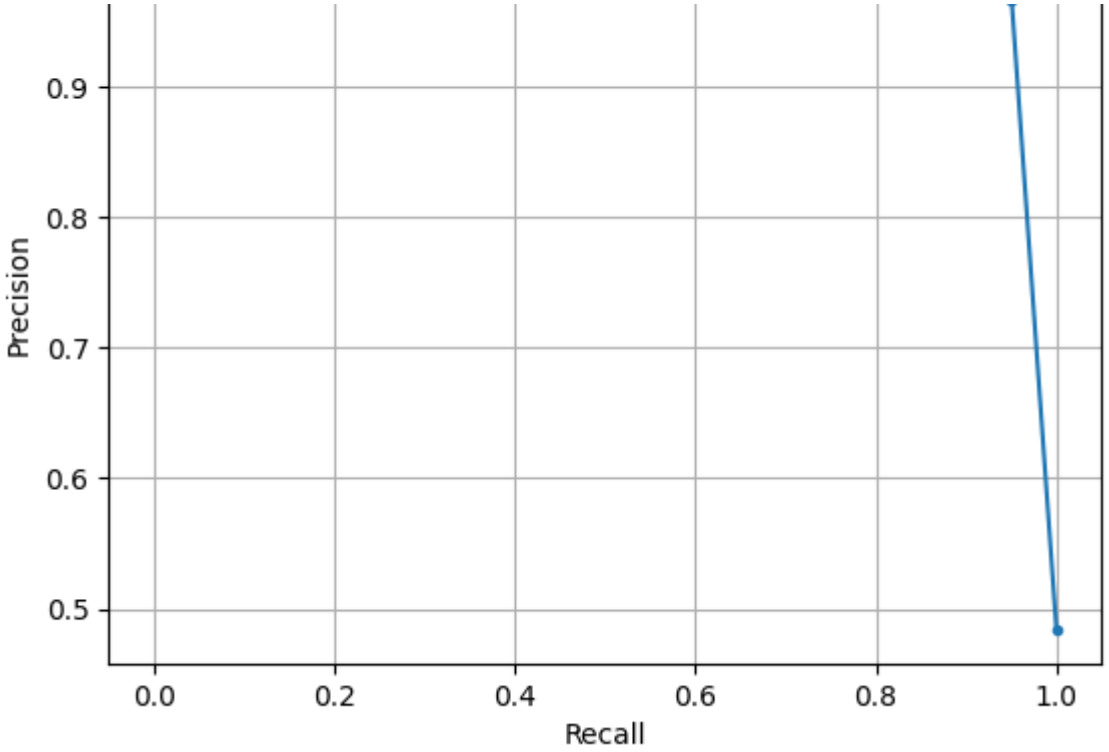

Blobs - Gradient Boost - learning_rate: 0.1
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

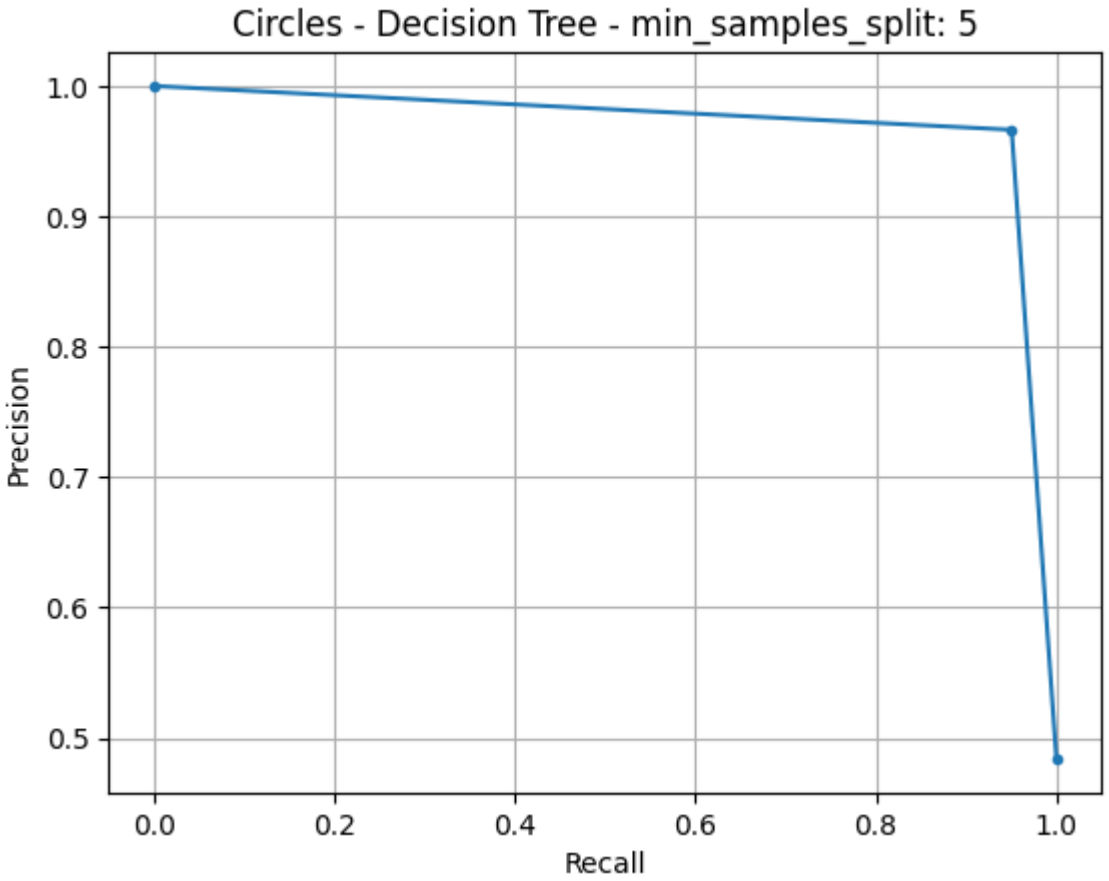

Blobs - Gradient Boost - learning_rate: 0.05
1.0
0.9
0.8
0.7
0.6
0.5
Precision

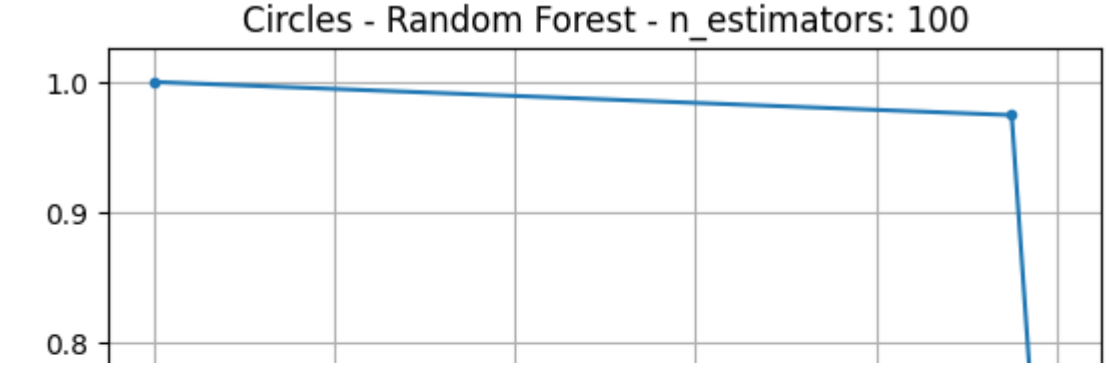

0.0
0.2
0.4
0.6
0.8
1.0
Recall
Circles - Decision Tree - max_depth: None
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

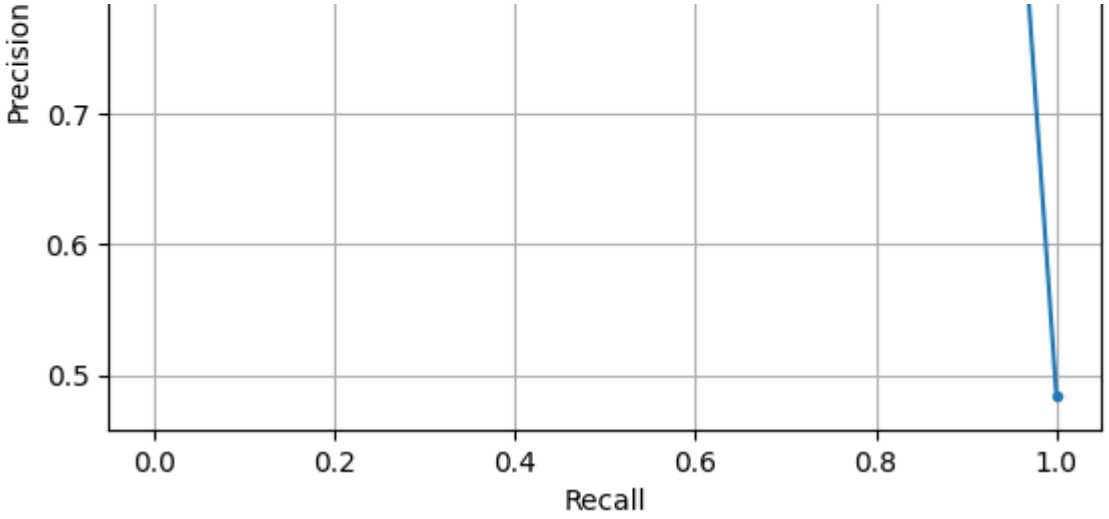

Circles - Decision Tree - max_depth: 3
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

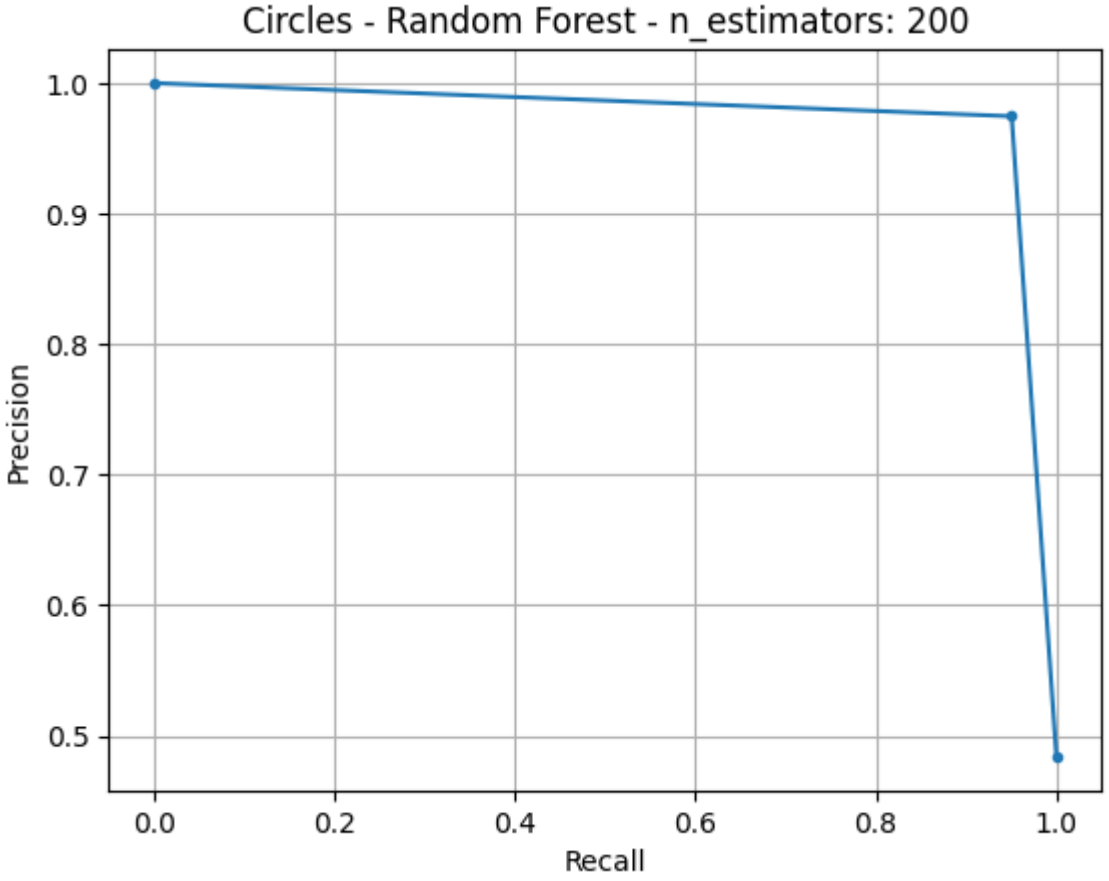

Circles - Decision Tree - min_samples_split: 2
1.0

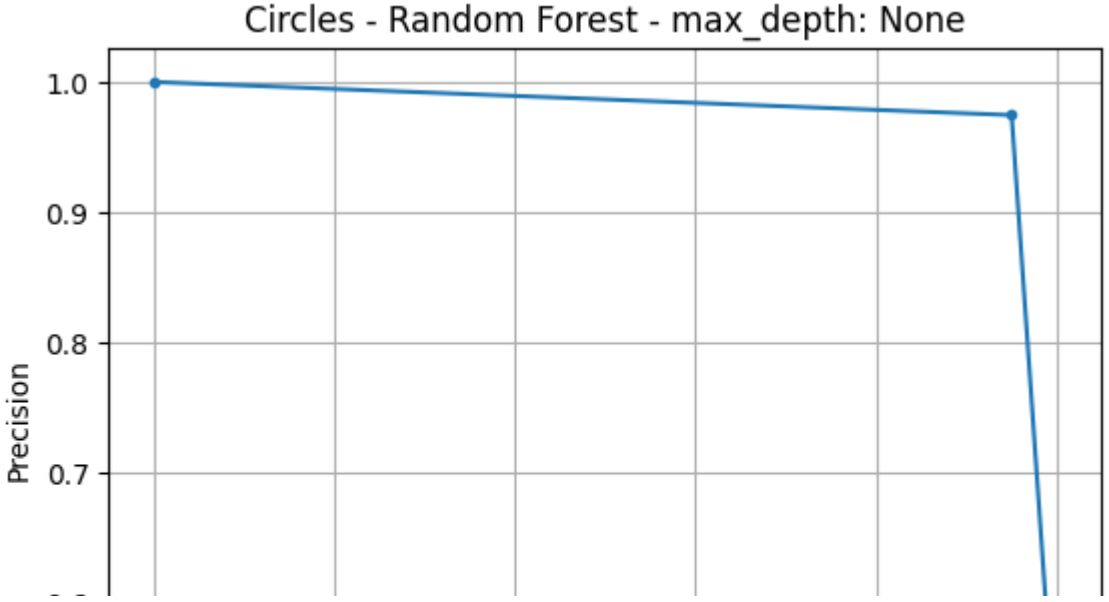

0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

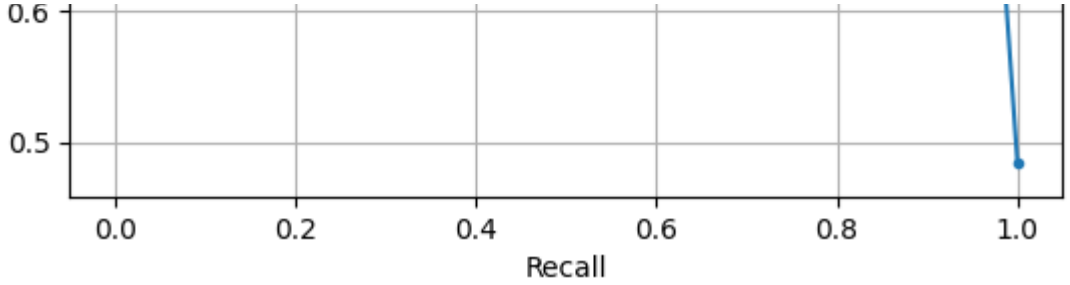

Circles - Decision Tree - min_samples_split: 5
1.0
0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

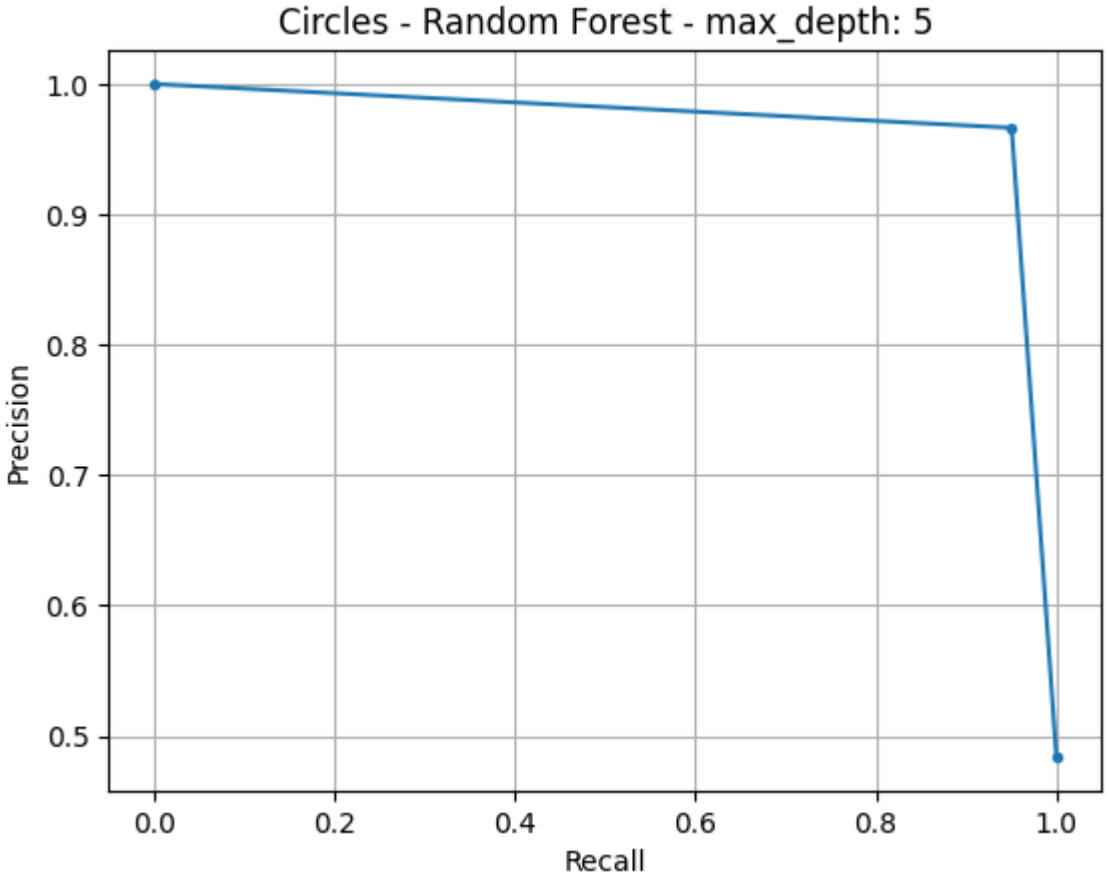

Circles - Random Forest - n_estimators: 100
1.0
0.9
0.8

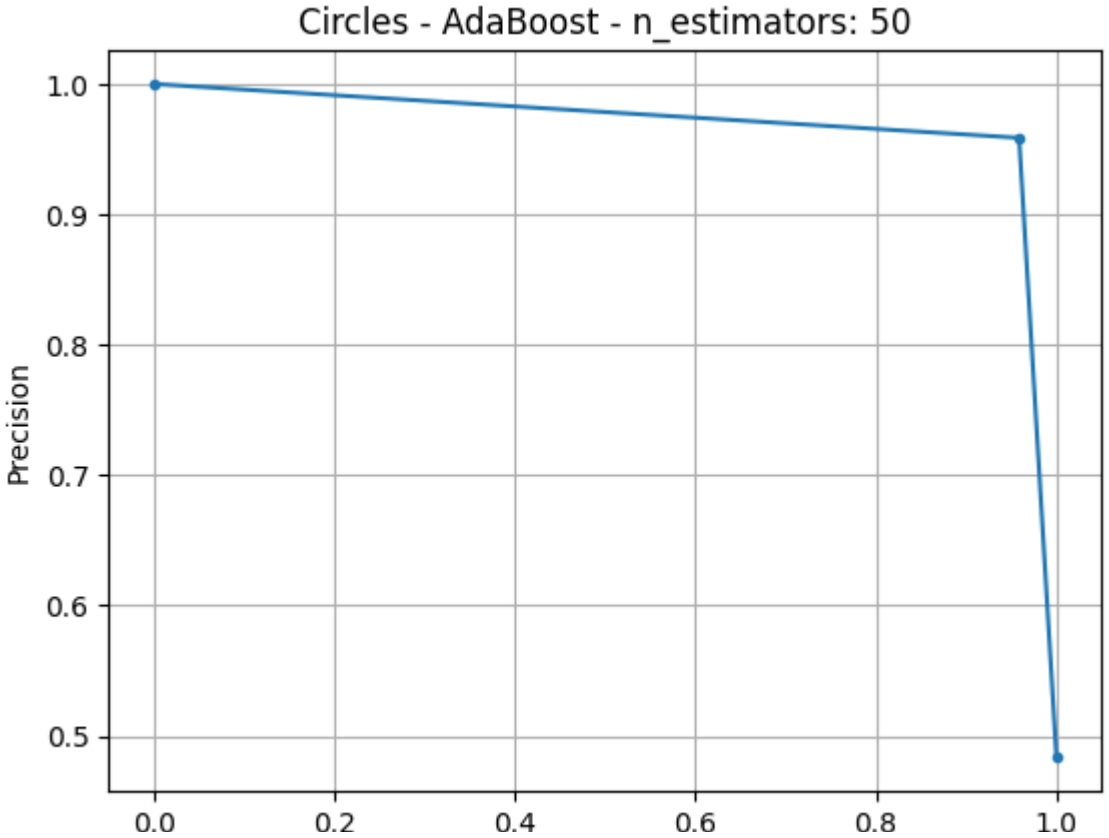

Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

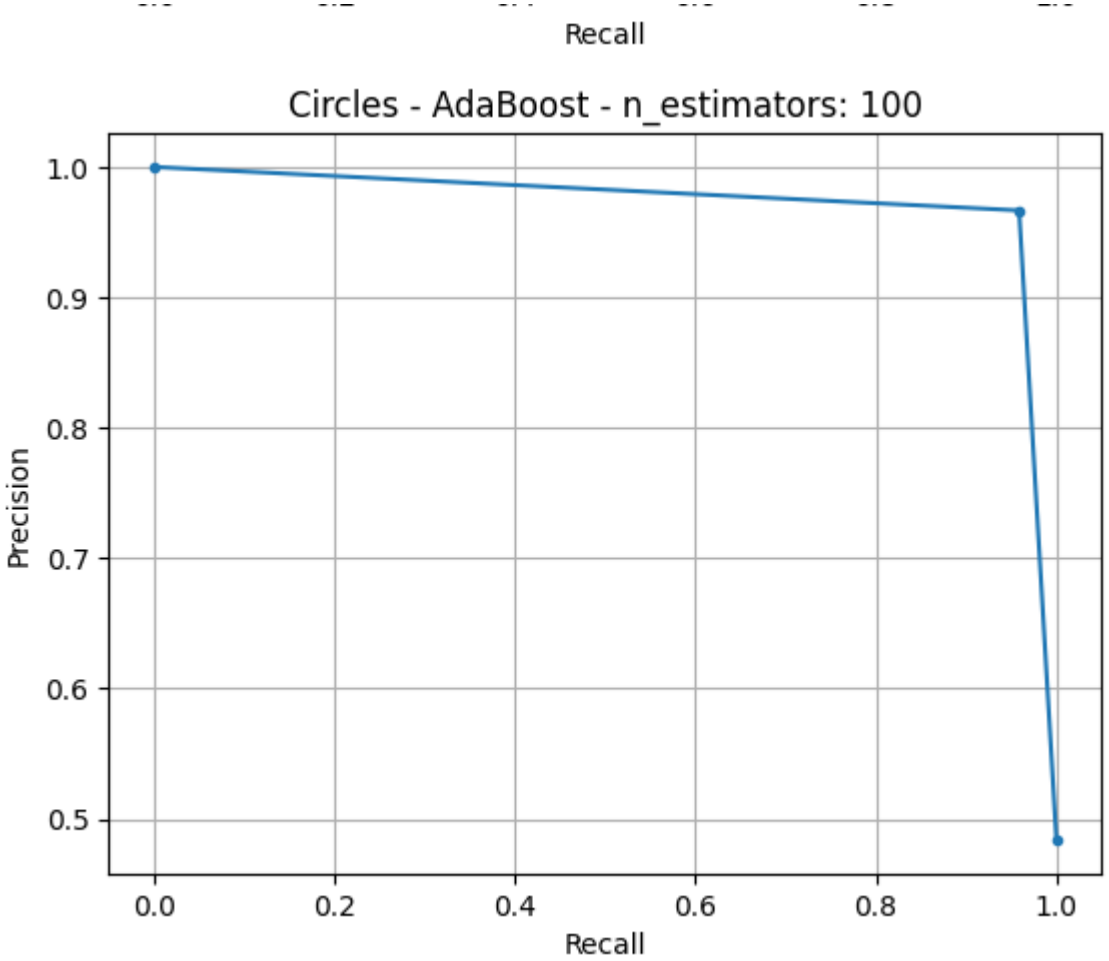

Circles - Random Forest - n_estimators: 200
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

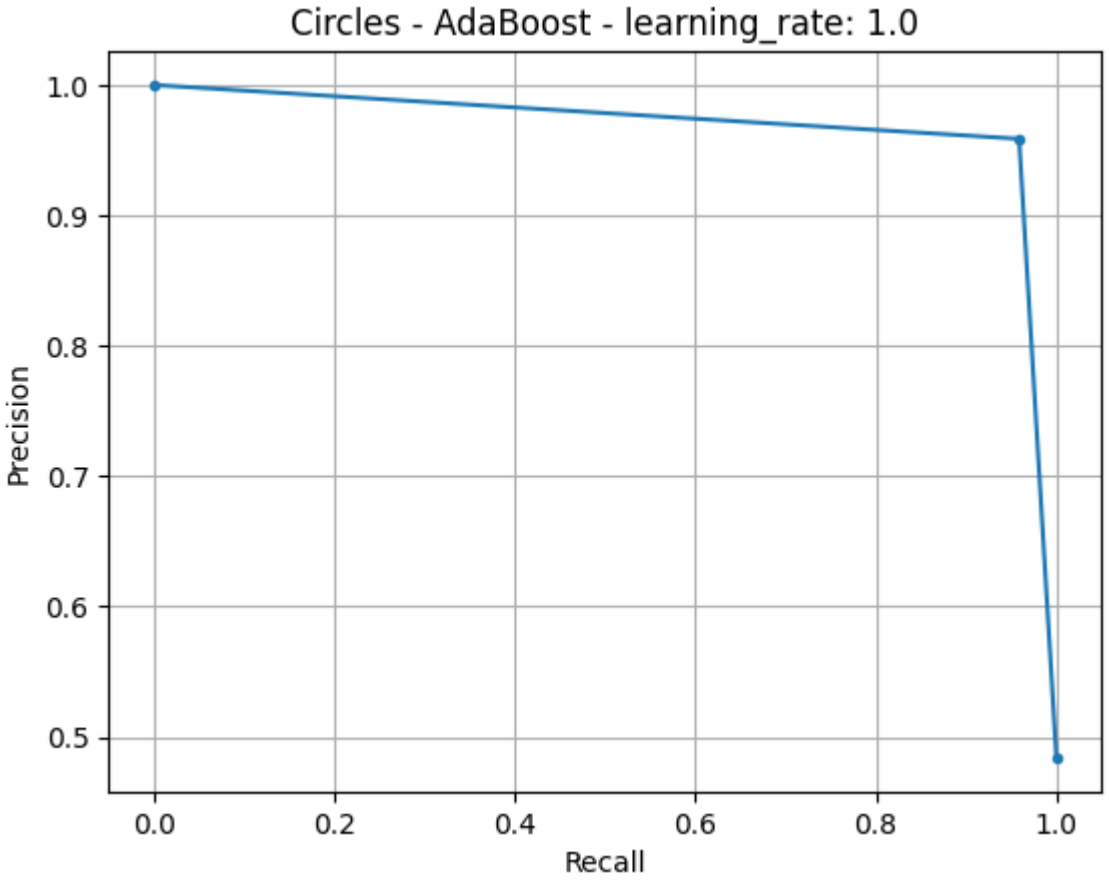

Circles - Random Forest - max_depth: None
Precision
1.0
0.9
0.8
0.7

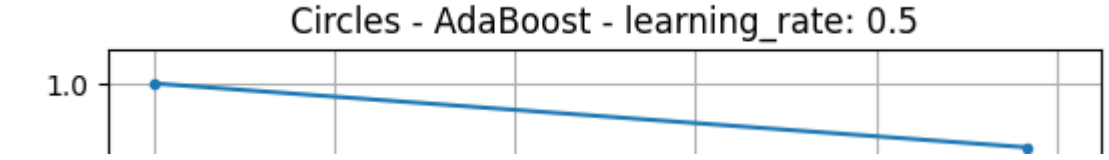

0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

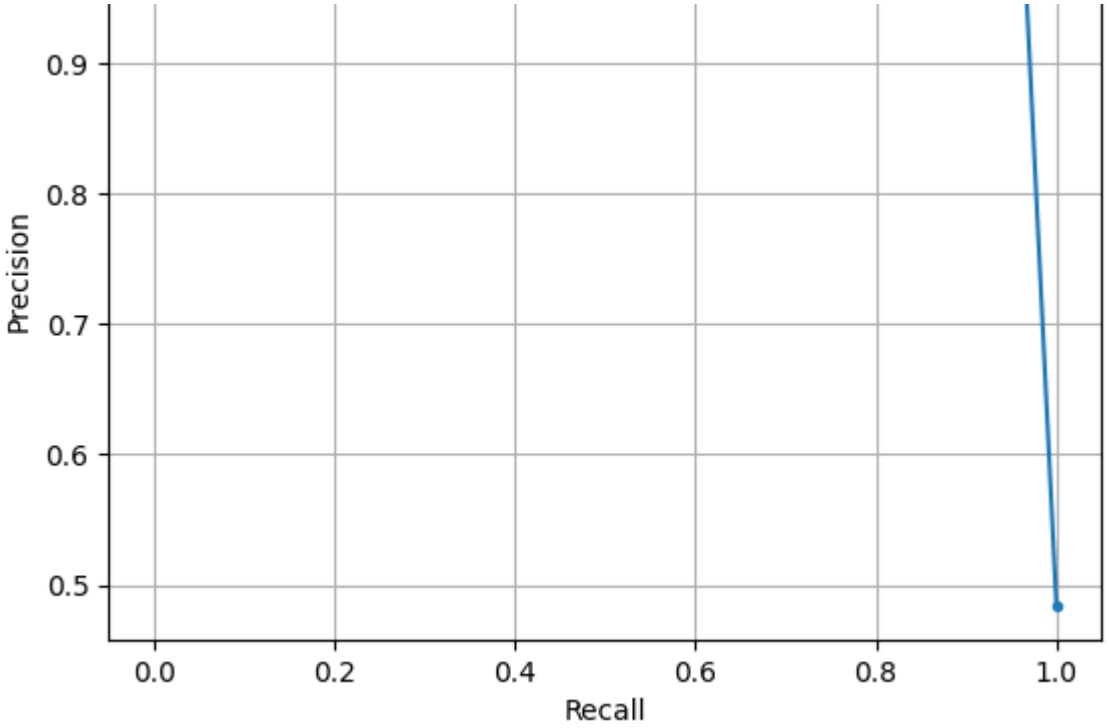

Circles - Random Forest - max_depth: 5
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

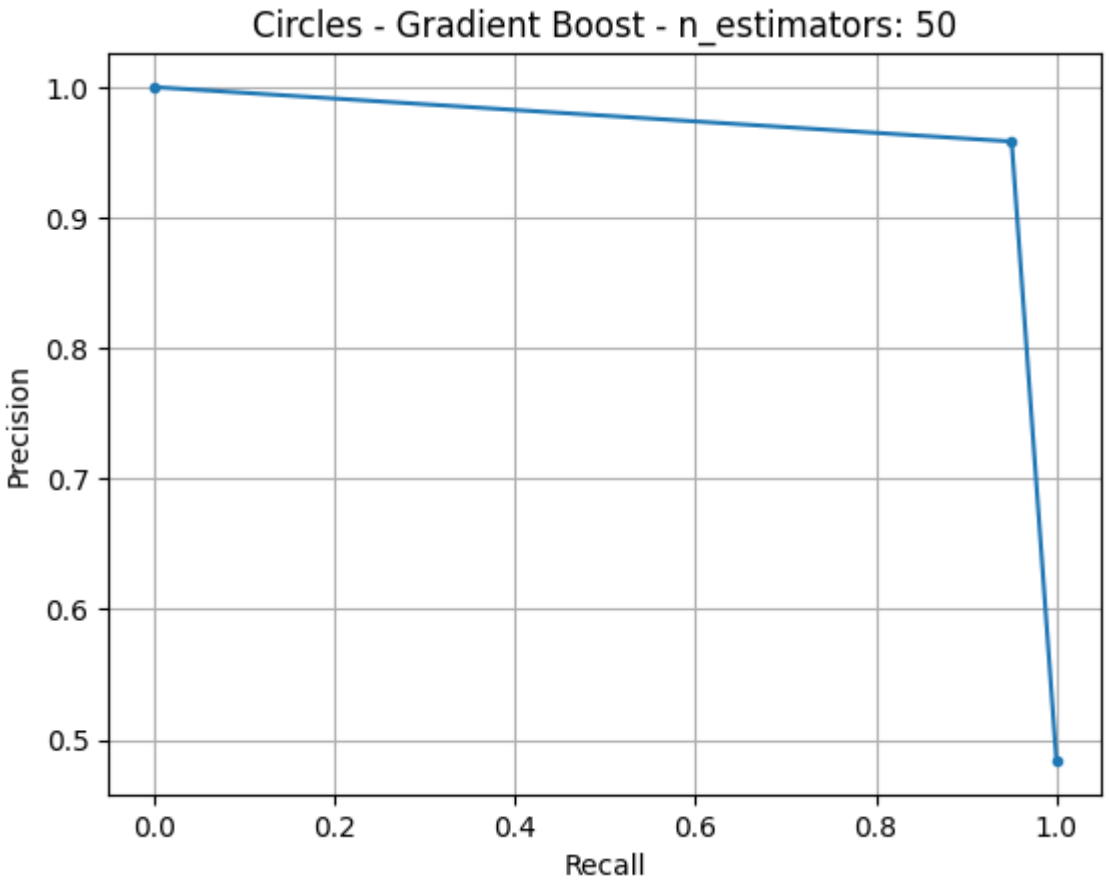

Circles - AdaBoost - n_estimators: 50
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0

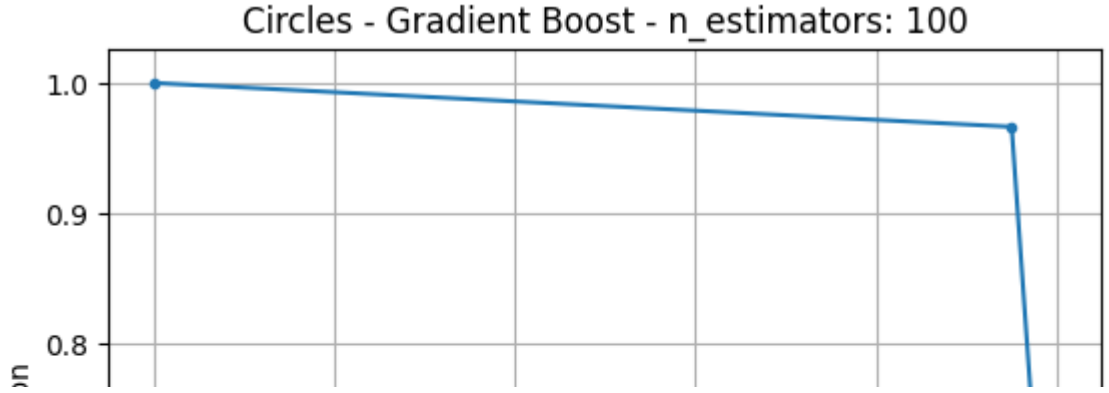

Recall
Circles - AdaBoost - n_estimators: 100
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

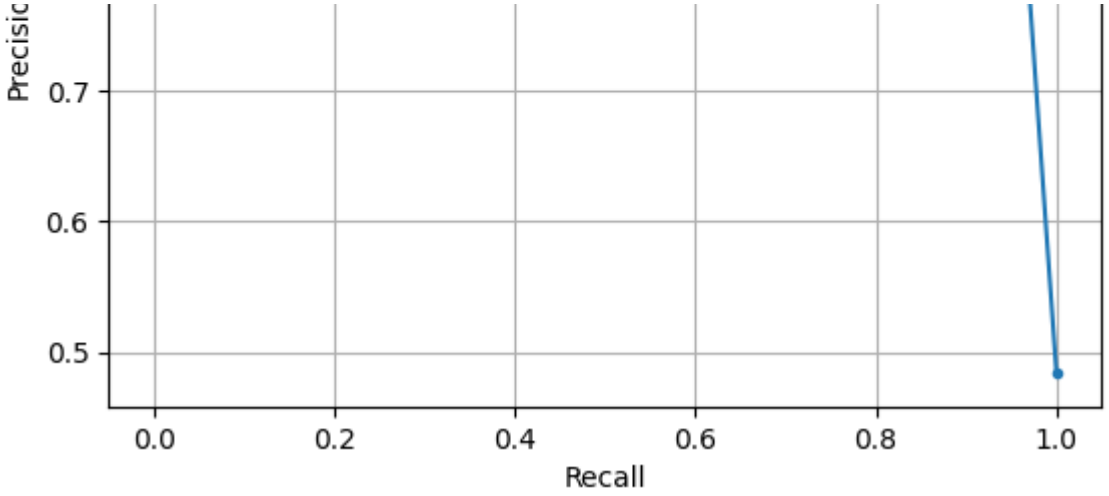

Circles - AdaBoost - learning_rate: 1.0
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

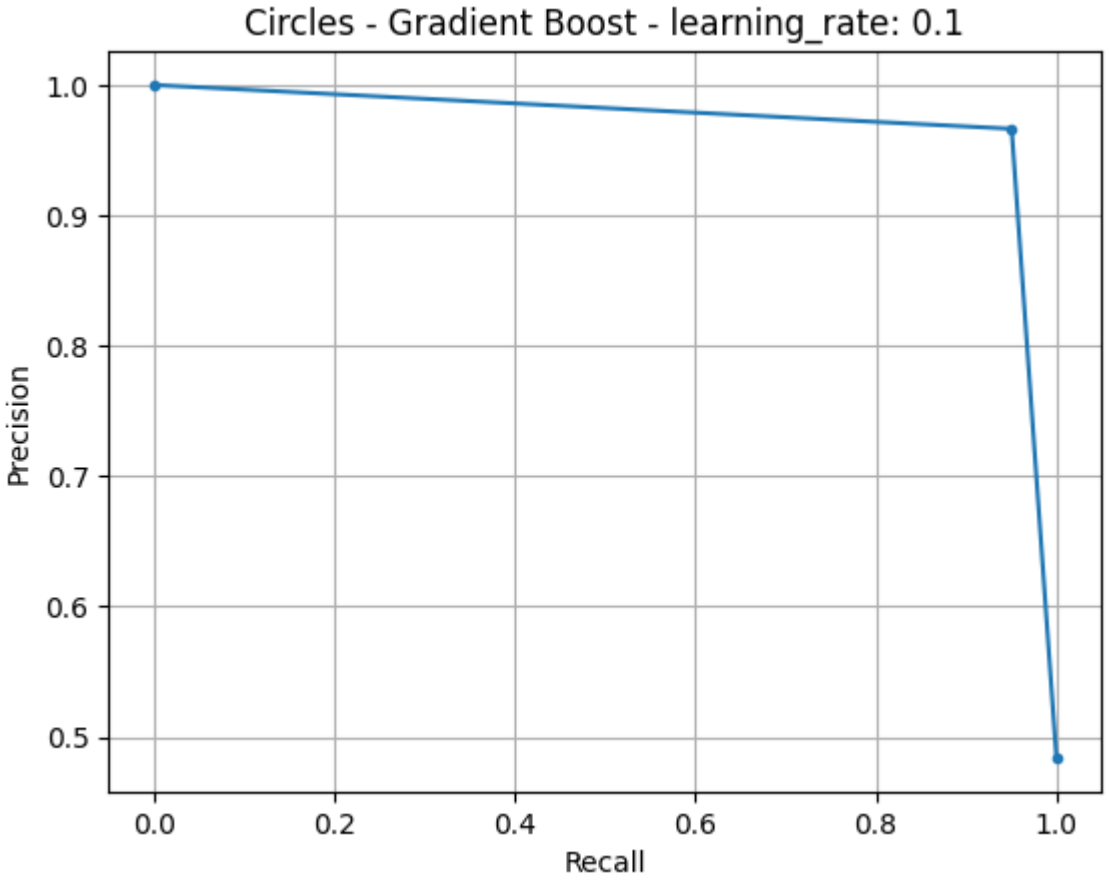

Circles - AdaBoost - learning_rate: 0.5
1.0

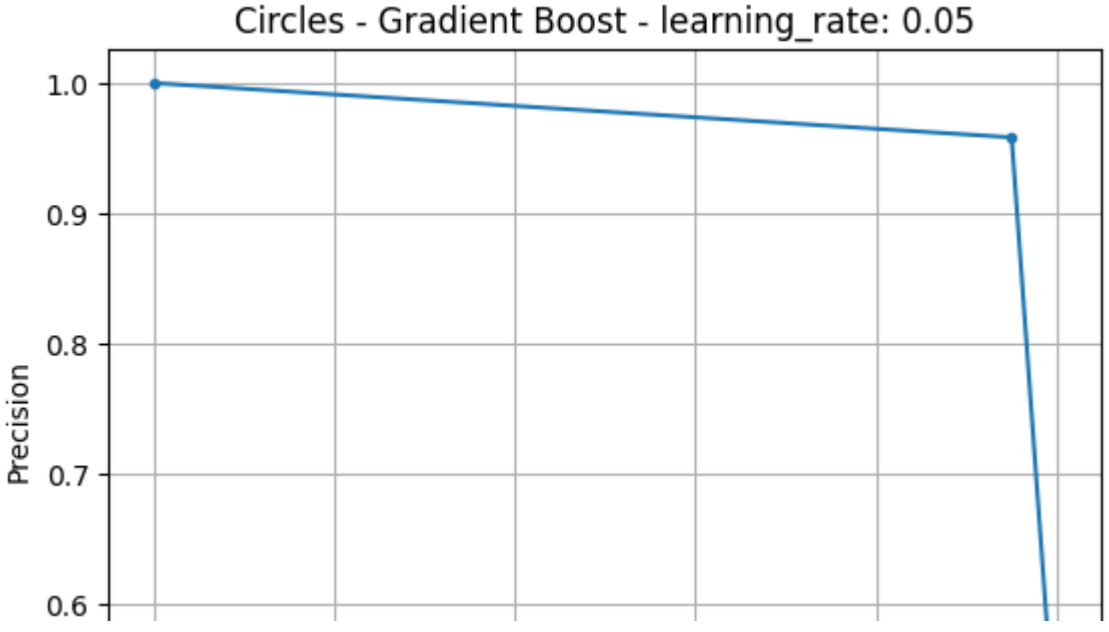

0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

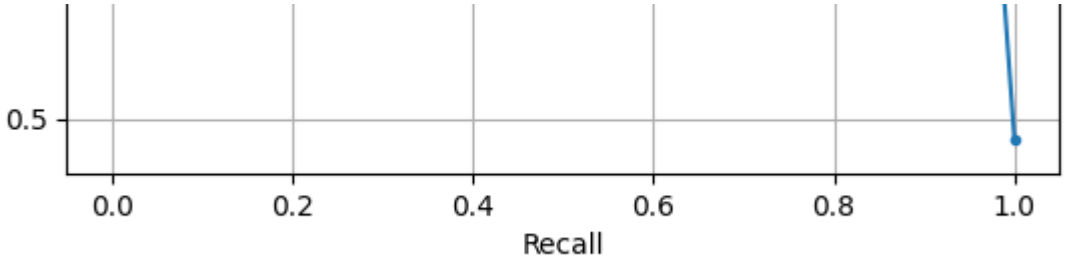

Circles - Gradient Boost - n_estimators: 50
1.0
0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

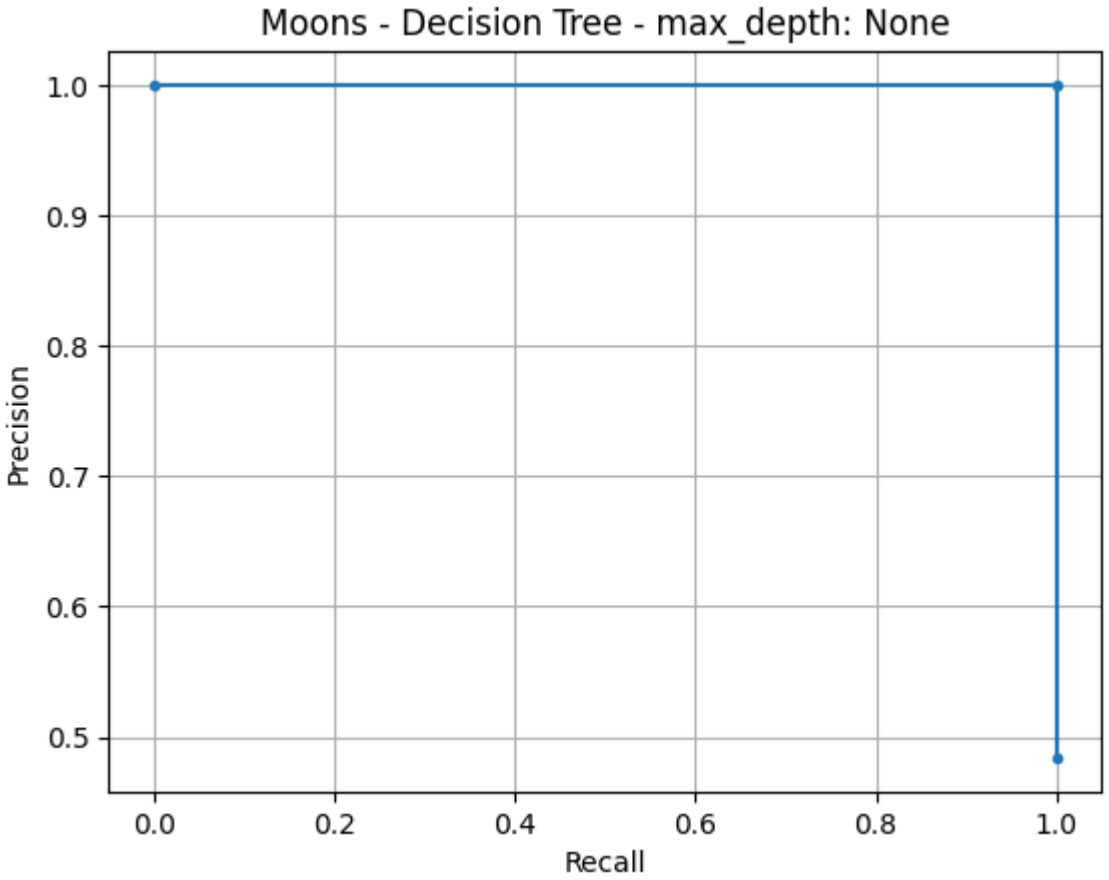

Circles - Gradient Boost - n_estimators: 100
1.0
0.9
0.8

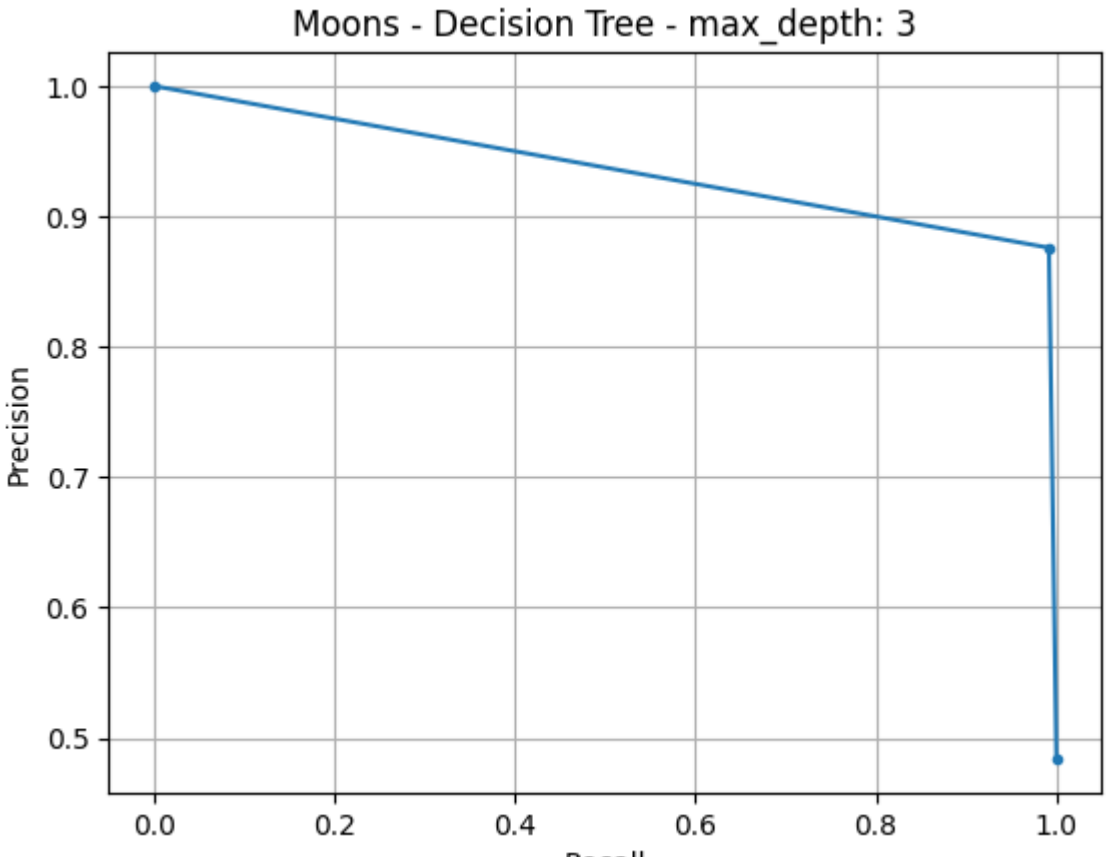

Precisic
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall


Circles - Gradient Boost - learning_rate: 0.1
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

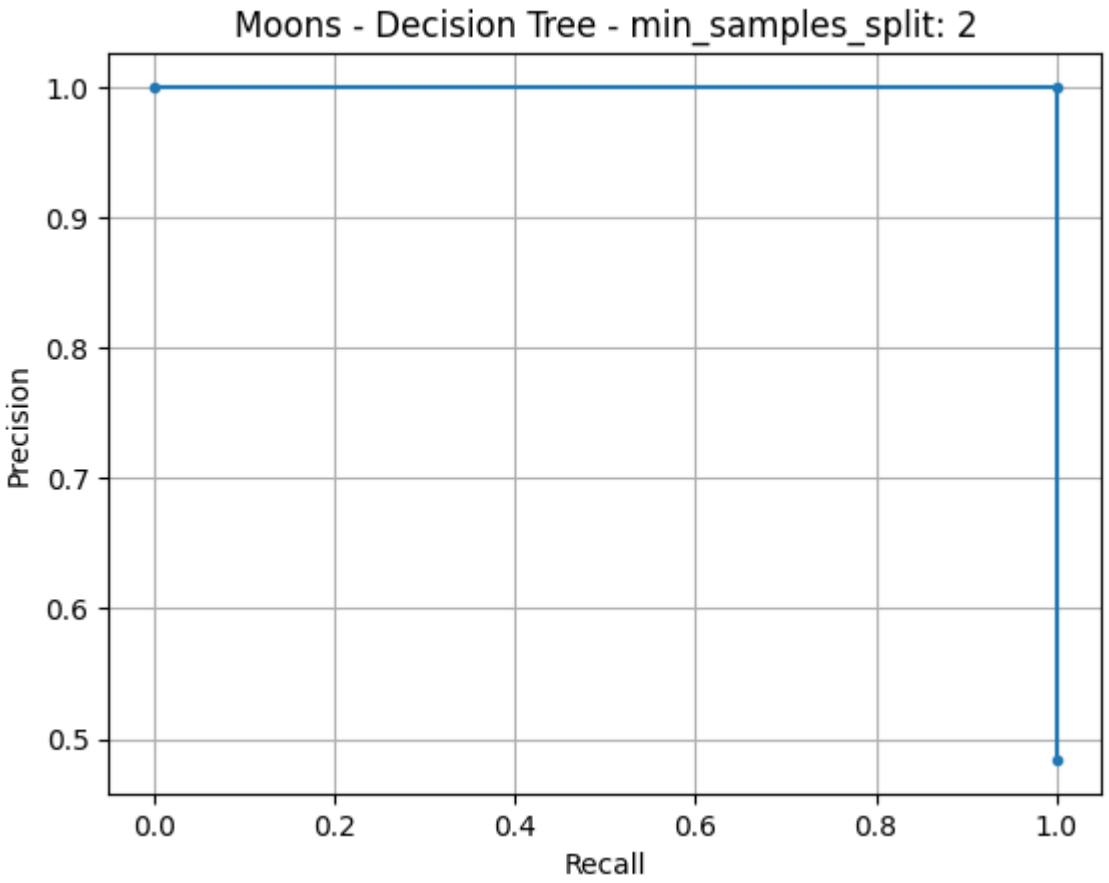

Circles - Gradient Boost - learning_rate: 0.05
Precision
1.0
0.9
0.8
0.7
0.6

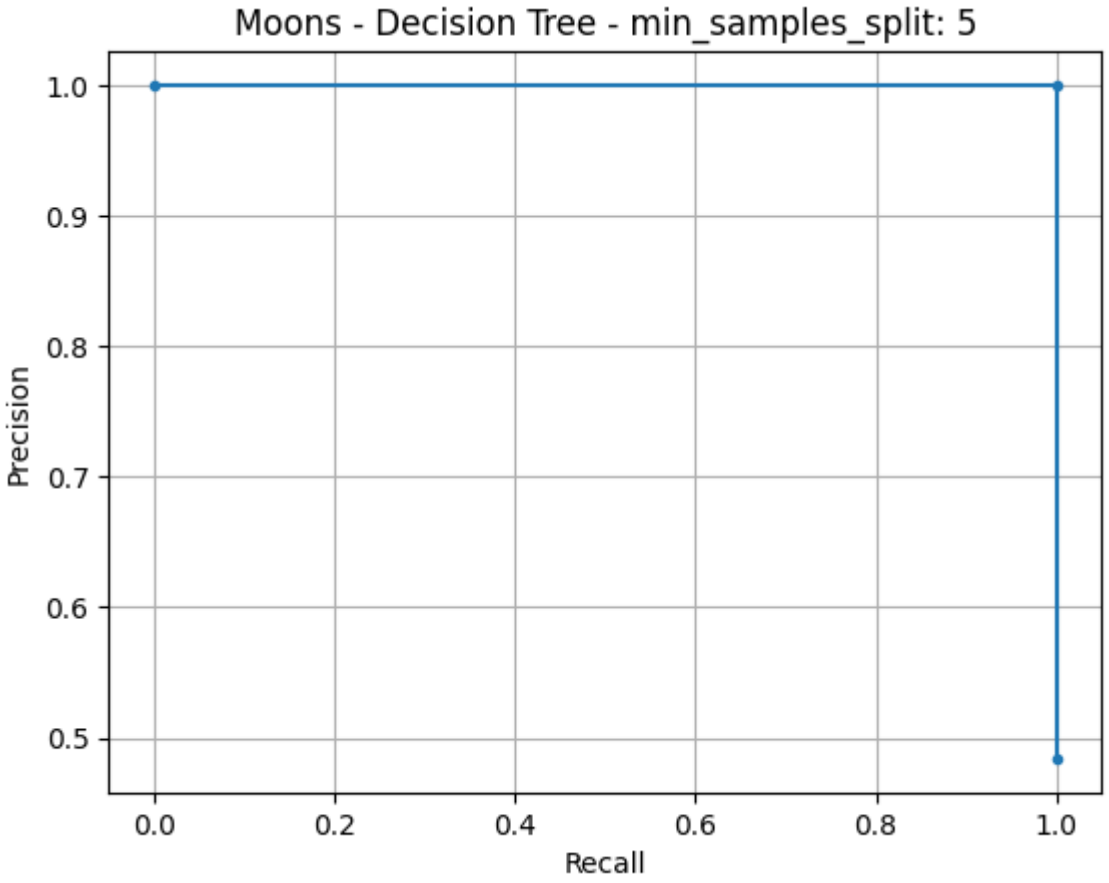

0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall


Moons - Decision Tree - max_depth: None
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

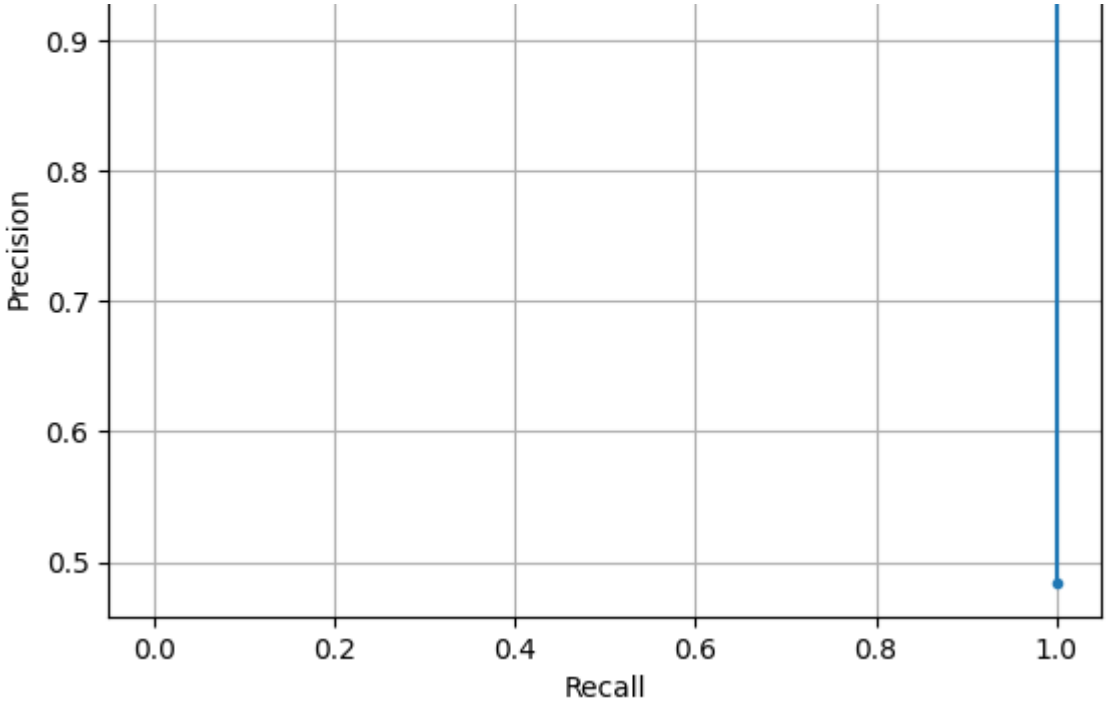

Moons - Decision Tree - max_depth: 3
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0

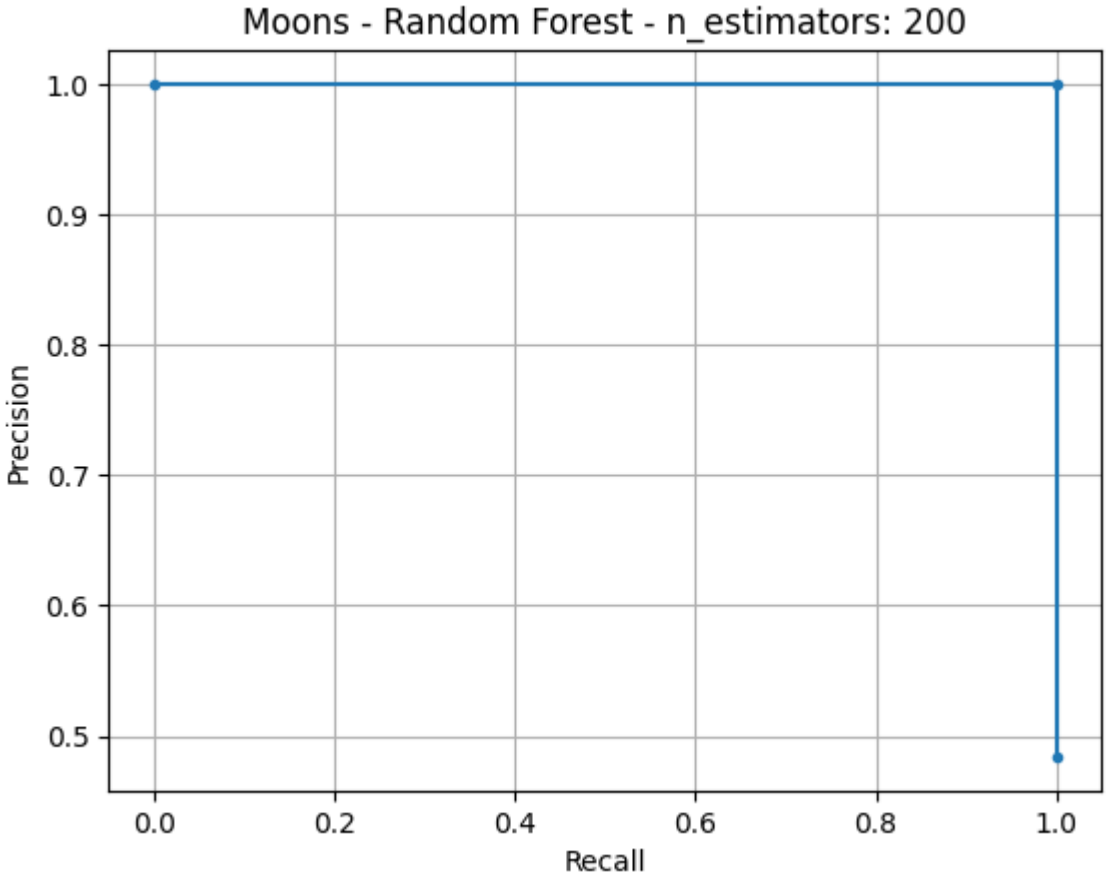
Recall

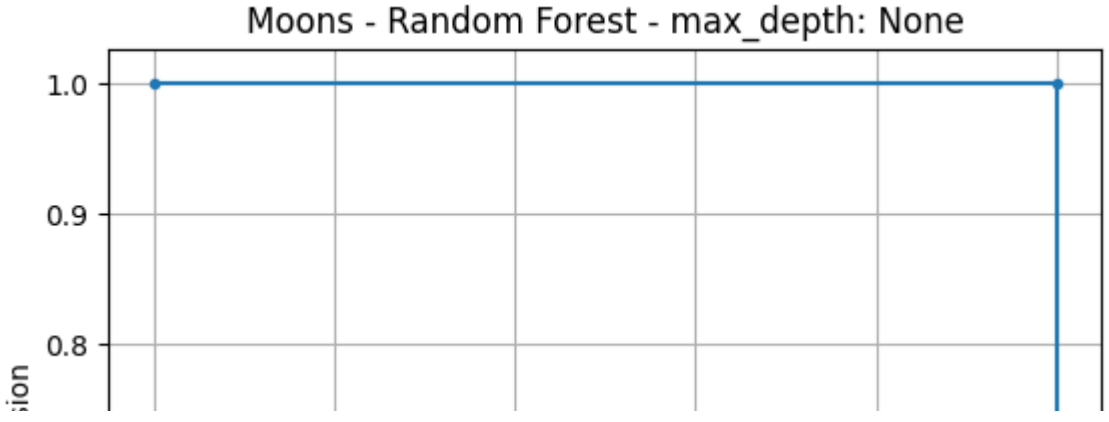

Moons - Decision Tree - min_samples_split: 2
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

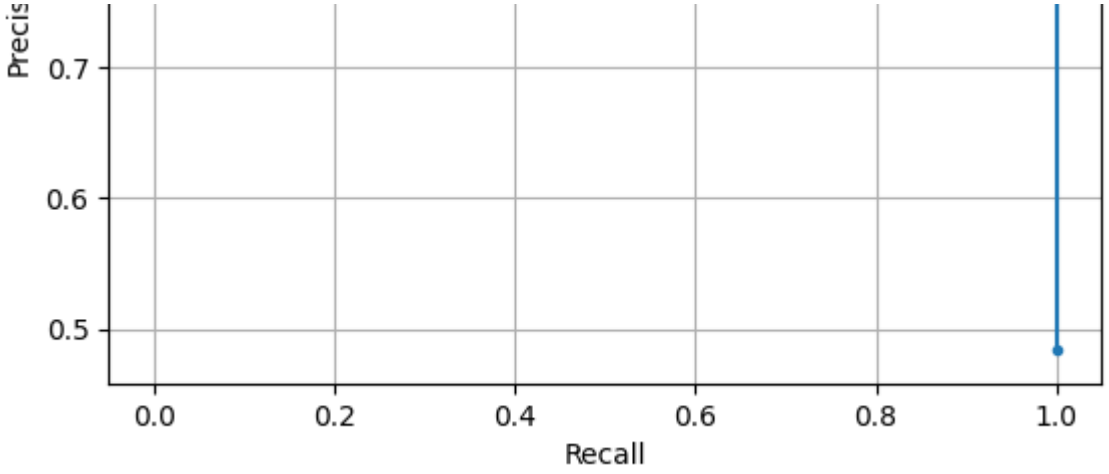

Moons - Decision Tree - min_samples_split: 5
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

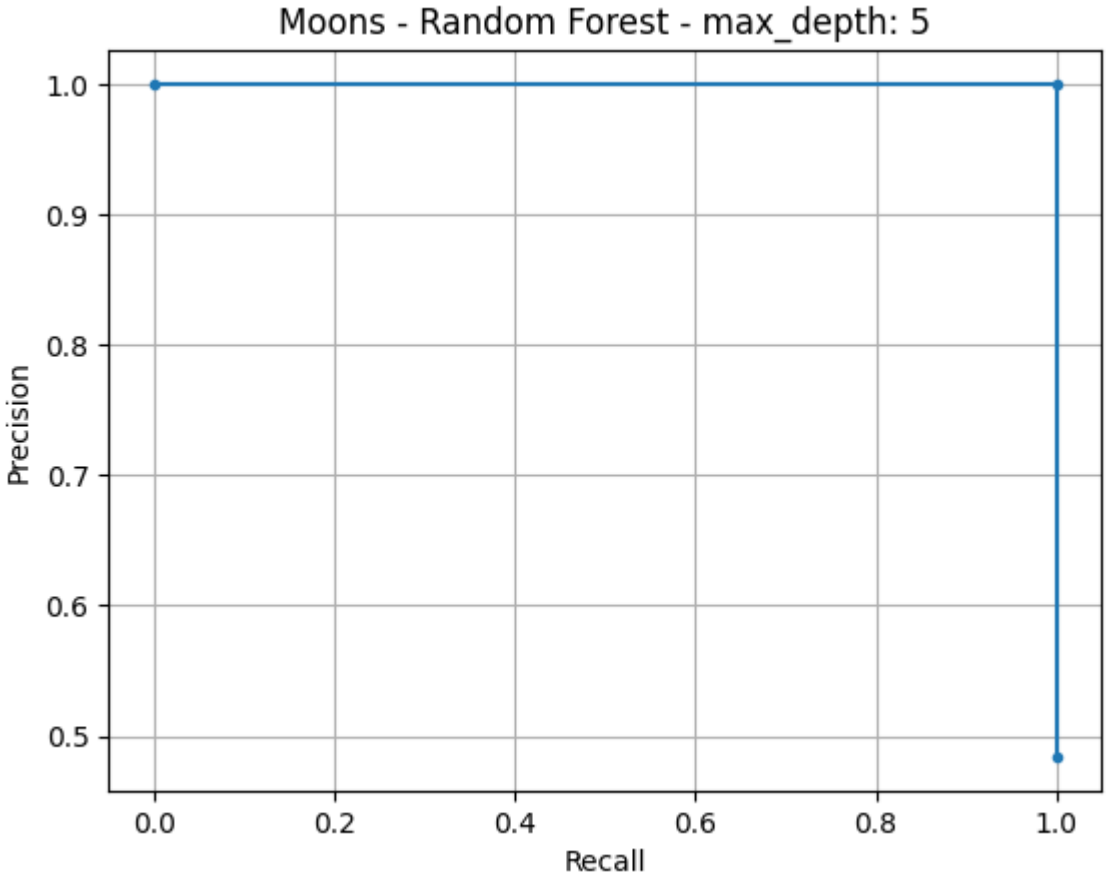

Moons - Random Forest - n_estimators: 100
1.0

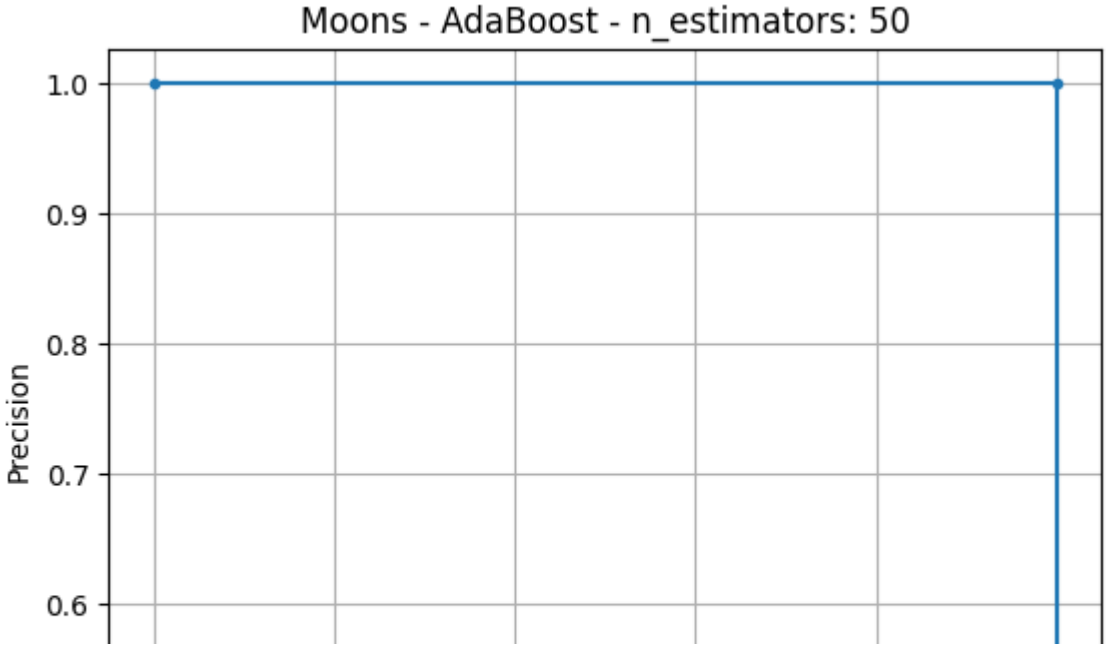

0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

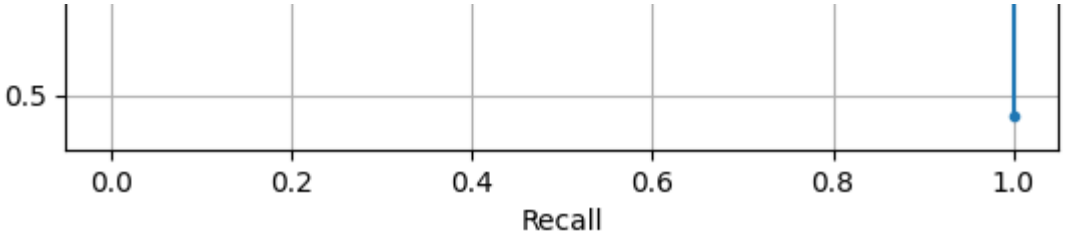

Moons - Random Forest - n_estimators: 200
1.0
0.9
0.8
Precision
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

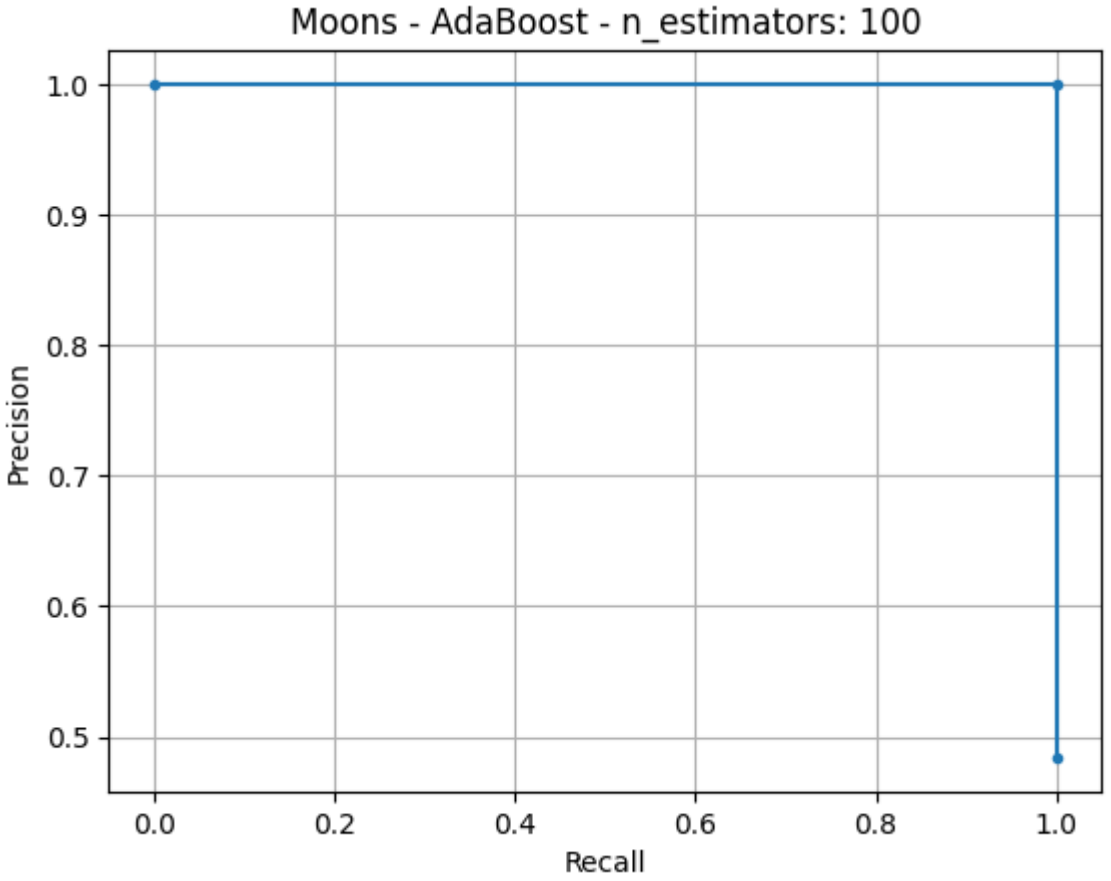

Moons - Random Forest - max_depth: None
1.0
0.9
0.8

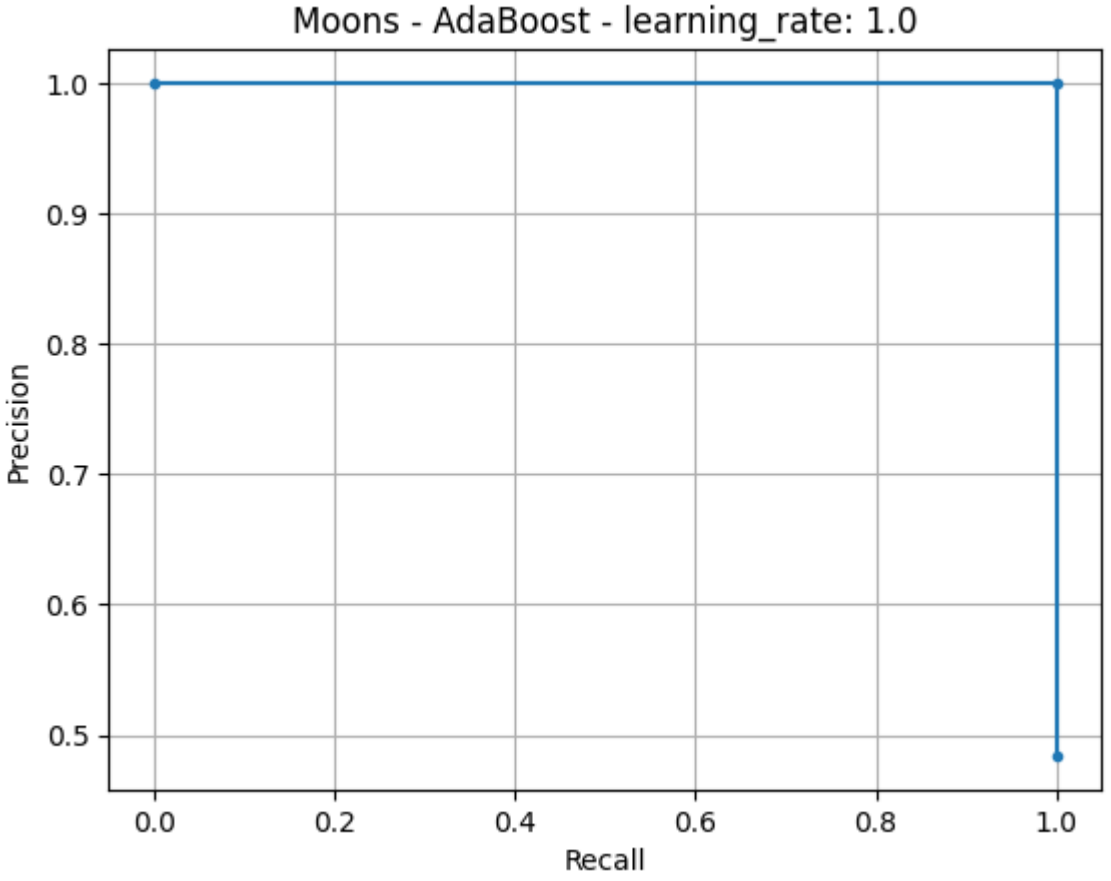

Precis
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

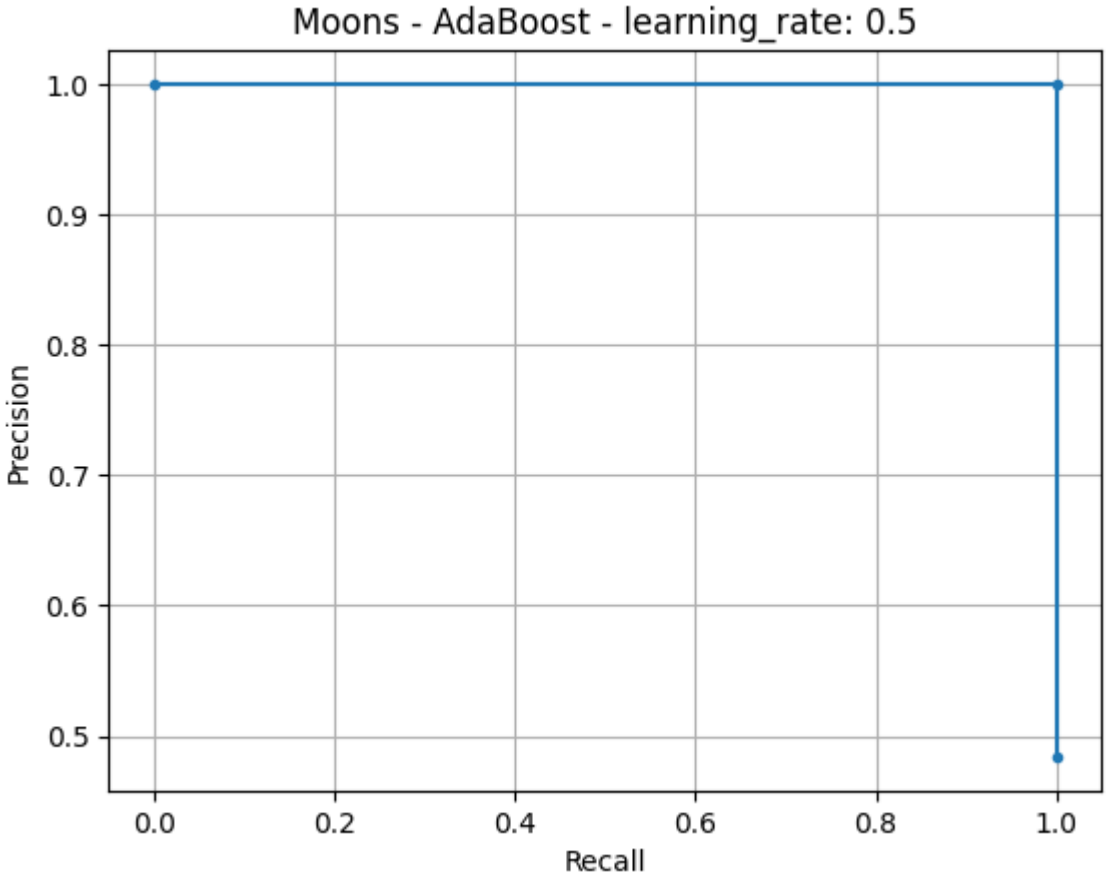

Moons - Random Forest - max_depth: 5
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

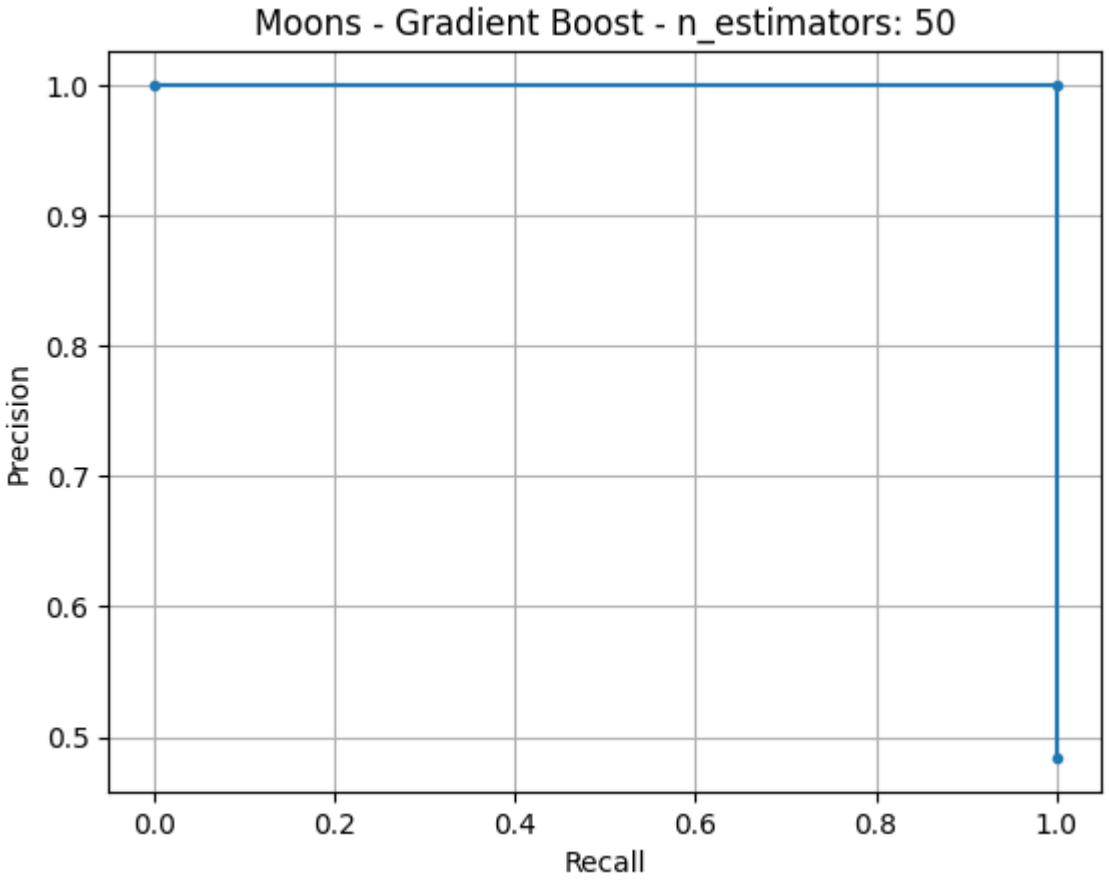

Moons - AdaBoost - n_estimators: 50
1.0
0.9
0.8
0.7
0.6
Precision

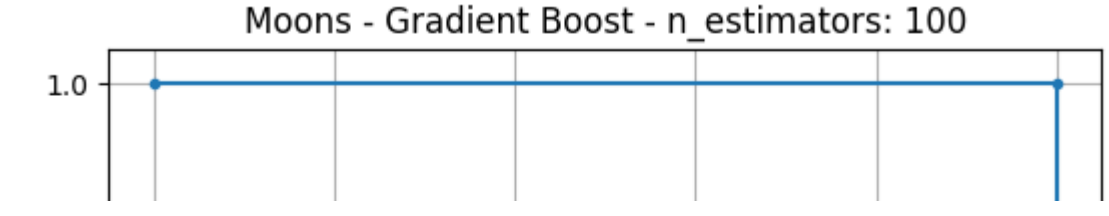

0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

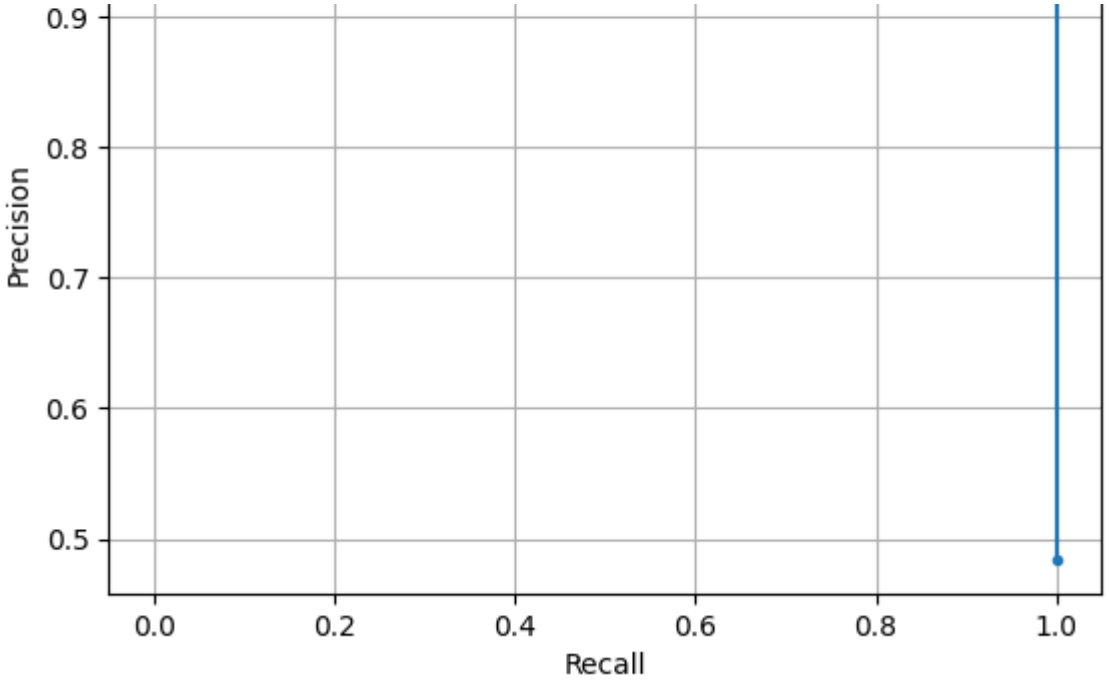

Moons - AdaBoost - n_estimators: 100
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

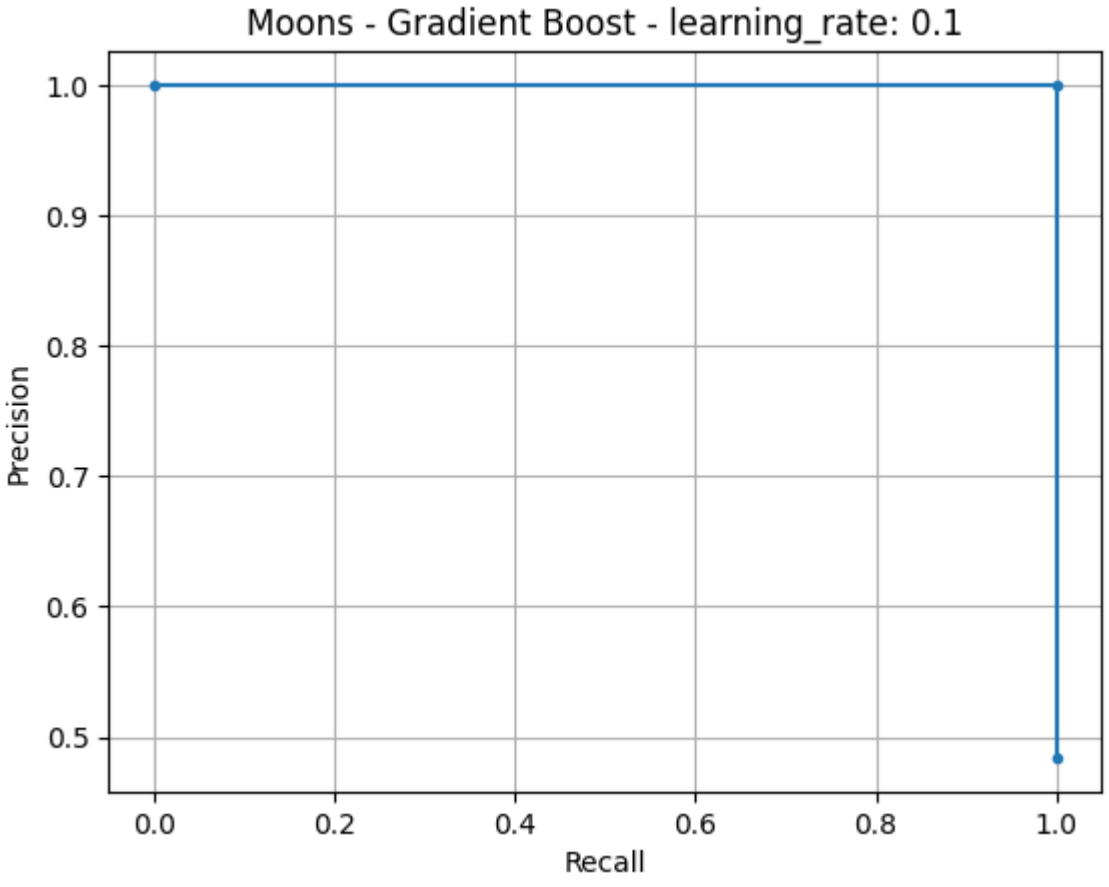

Moons - AdaBoost - learning_rate: 1.0
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Recall

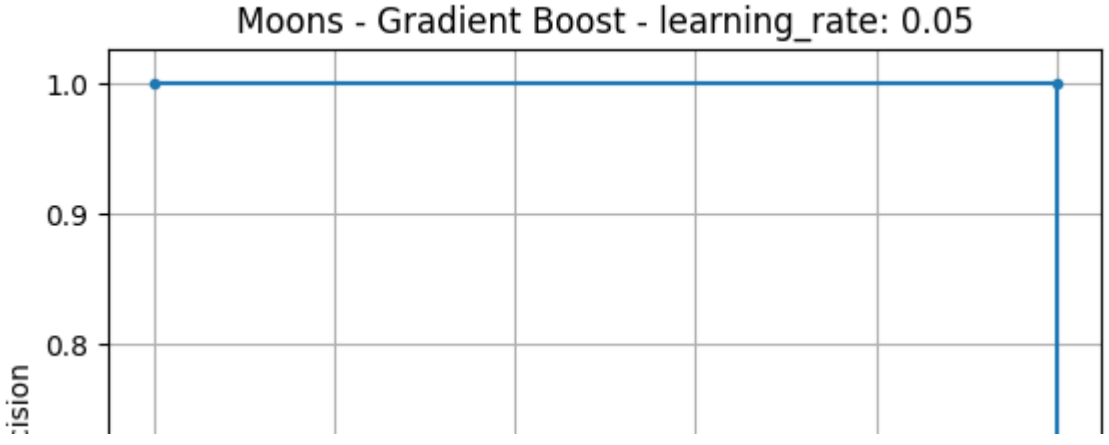

Moons - AdaBoost - learning_rate: 0.5
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

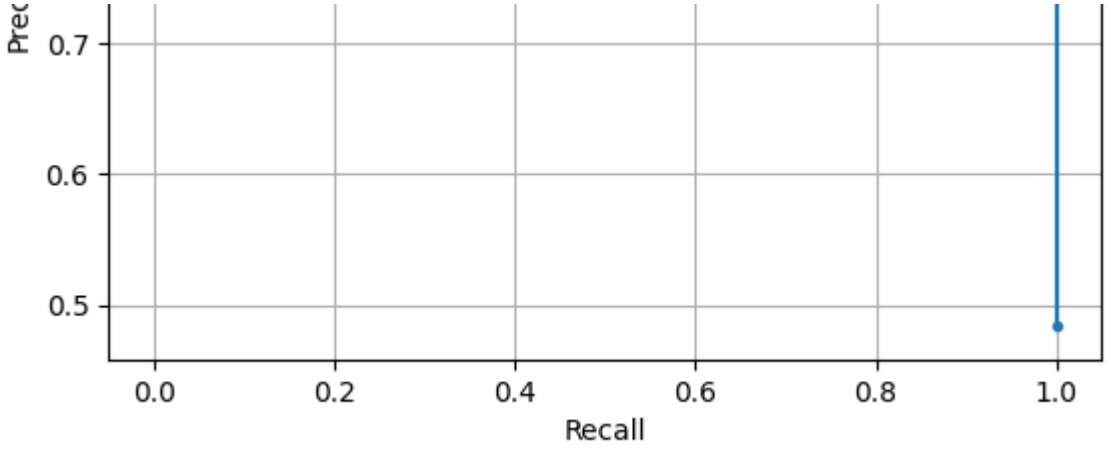

Moons - Gradient Boost - n_estimators: 50
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0


Moons - Gradient Boost - n_estimators: 100
1.0


0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall


Moons - Gradient Boost - learning_rate: 0.1
1.0
0.9
0.8
0.7
0.6
0.5
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall


Moons - Gradient Boost - learning_rate: 0.05
1.0
0.9
0.8




```
# Print the results
print("\nResults:")
for dataset_name, dataset_results in results.items():
    print(f"\nDataset: {dataset_name}")
    for model_name, params in dataset_results.items():
        print(f"\n{model_name}:")
        for param_config, metrics in params.items():
            print(f"{param_config}: {metrics}")
```

```
Results:

Dataset: Blobs

Decision Tree:
('max_depth', None): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1
('max_depth', 3): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1-sc
('min_samples_split', 2): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0
('min_samples_split', 5): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0

Random Forest:
('n_estimators', 100): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('n_estimators', 200): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('max_depth', None): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1
('max_depth', 5): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1-sc

AdaBoost:
('n_estimators', 50): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F
('n_estimators', 100): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('learning_rate', 1.0): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,
('learning_rate', 0.5): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,

Gradient Boost:
('n_estimators', 50): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F
('n_estimators', 100): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('learning_rate', 0.1): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,
('learning_rate', 0.05): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,

Dataset: Circles

Decision Tree:
('max_depth', None): {'Accuracy': 0.964, 'Precision': 0.9642403697996919, '
('max_depth', 3): {'Accuracy': 0.82, 'Precision': 0.8393333333333334, 'Reca
```

```
('max_depth', 5): {'Accuracy': 0.82, 'Precision': 0.8333333333333334, 'Reca
('min_samples_split', 2): {'Accuracy': 0.96, 'Precision': 0.960097504650715
('min_samples_split', 5): {'Accuracy': 0.96, 'Precision': 0.960097504650715

Random Forest:
('n_estimators', 100): {'Accuracy': 0.964, 'Precision': 0.9642403697996919,
('n_estimators', 200): {'Accuracy': 0.964, 'Precision': 0.9642403697996919,
('max_depth', None): {'Accuracy': 0.964, 'Precision': 0.9642403697996919, '
('max_depth', 5): {'Accuracy': 0.96, 'Precision': 0.9600975046507153, 'Reca

AdaBoost:
('n_estimators', 50): {'Accuracy': 0.96, 'Precision': 0.96, 'Recall': 0.96,
('n_estimators', 100): {'Accuracy': 0.964, 'Precision': 0.9640205128205128,
('learning_rate', 1.0): {'Accuracy': 0.96, 'Precision': 0.96, 'Recall': 0.9
('learning_rate', 0.5): {'Accuracy': 0.96, 'Precision': 0.9601382753985019,

Gradient Boost:
('n_estimators', 50): {'Accuracy': 0.956, 'Precision': 0.9560179487179488,
('n_estimators', 100): {'Accuracy': 0.96, 'Precision': 0.9600975046507153,
('learning_rate', 0.1): {'Accuracy': 0.96, 'Precision': 0.9600975046507153,
('learning_rate', 0.05): {'Accuracy': 0.956, 'Precision': 0.956017948717948

Dataset: Moons

Decision Tree:
```

```
import numpy as np
from sklearn.datasets import (fetch_olivetti_faces, fetch_20newsgroups, fetch_l
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Gradie
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s
import matplotlib.pyplot as plt

# Define datasets
datasets = {
    "Olivetti Faces": fetch_olivetti_faces(),
}

# Define parameter configurations for each classifier
model_params = {
    "Decision Tree": {
        "max_depth": [3, 5],
        "min_samples_split": [2, 5]
        },
    "Random Forest": {
        "n_estimators": [100, 200],
        "max_depth": [3, 5]
        },
    "AdaBoost": {
        "n_estimators": [50, 100],
        "learning_rate": [1.0, 0.5]
        },
    "Gradient Boost": {
        "n_estimators": [50, 100],
        "learning_rate": [0.1, 0.05]
        }
}

import warnings
warnings.filterwarnings('ignore')


# Initialize a dictionary to store the results
results = {}

# Loop through each dataset
for dataset_name, dataset in datasets.items():
    print(f"\n\nDataset: {dataset_name}")
    X = dataset.data
    y = dataset.target

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

    results[dataset_name] = {}

    # Loop through each classifier and its parameter configurations
```

```
    for model_name, params in model_params.items():
        print(f"\nTraining {model_name}...")
        results[dataset_name][model_name] = {}

        for param_name, param_values in params.items():
            for param_value in param_values:
                # Create and train the classifier with the current parameter con
                if model_name == "Decision Tree":
                    classifier = DecisionTreeClassifier(**{param_name: param_valu
                elif model_name == "Random Forest":
                    classifier = RandomForestClassifier(**{param_name: param_valu
                elif model_name == "AdaBoost":
                    classifier = AdaBoostClassifier(**{param_name: param_value})
                elif model_name == "Gradient Boost":
                    classifier = GradientBoostingClassifier(**{param_name: param_
                else:
                    raise ValueError(f"Unknown model name: {model_name}")

                classifier.fit(X_train, y_train)

                # Make predictions on the test set
                y_pred = classifier.predict(X_test)

                # Calculate evaluation metrics
                accuracy = accuracy_score(y_test, y_pred)
                precision = precision_score(y_test, y_pred, average="weighted")
                recall = recall_score(y_test, y_pred, average="weighted")
                f1 = f1_score(y_test, y_pred, average="weighted")

                # Calculate precision-recall curve
                precision_curve, recall_curve, _ = precision_recall_curve(y_test
                                                                          pos_la
                average_precision = np.mean(precision_curve)
                average_recall = np.mean(recall_curve)

                # Store the results
                results[dataset_name][model_name][(param_name, param_value)] = {
                    "Accuracy": accuracy,
                    "Precision": precision,
                    "Recall": recall,
                    "F1-score": f1,
                    "Average Precision": average_precision,
                    "Average Recall": average_recall
                }
                print(f"{param_name}: {param_value}, Accuracy: {accuracy:.2f}, Pr
                      f"Recall: {recall:.2f}, F1-score: {f1:.2f}, Average Precis
                      f"Average Recall: {average_recall:.2f}")

                # Plot precision-recall curve and save figure
                plt.figure()
                plt.plot(recall_curve, precision_curve, marker='.')
                plt.title(f'{dataset_name} - {model_name} - {param_name}: {param_
                plt.xlabel('Recall')
                plt.ylabel('Precision')
```

```
            plt.grid(True)
            plt.savefig(f'{dataset_name}_{model_name}_{param_name}_{param_va
```

```
Dataset: Olivetti Faces

Training Decision Tree...
max_depth: 3, Accuracy: 0.03, Precision: 0.01, Recall: 0.03, F1-score: 0.02
max_depth: 5, Accuracy: 0.08, Precision: 0.07, Recall: 0.08, F1-score: 0.07
min_samples_split: 2, Accuracy: 0.46, Precision: 0.47, Recall: 0.46, F1-sco
min_samples_split: 5, Accuracy: 0.45, Precision: 0.53, Recall: 0.45, F1-sco

Training Random Forest...
n_estimators: 100, Accuracy: 0.92, Precision: 0.96, Recall: 0.92, F1-score:
n_estimators: 200, Accuracy: 0.88, Precision: 0.93, Recall: 0.88, F1-score:
max_depth: 3, Accuracy: 0.39, Precision: 0.36, Recall: 0.39, F1-score: 0.34
max_depth: 5, Accuracy: 0.54, Precision: 0.53, Recall: 0.54, F1-score: 0.50

Training AdaBoost...
n_estimators: 50, Accuracy: 0.07, Precision: 0.11, Recall: 0.07, F1-score:
n_estimators: 100, Accuracy: 0.07, Precision: 0.10, Recall: 0.07, F1-score:
learning_rate: 1.0, Accuracy: 0.07, Precision: 0.11, Recall: 0.07, F1-score
learning_rate: 0.5, Accuracy: 0.22, Precision: 0.32, Recall: 0.22, F1-score

Training Gradient Boost...
```

```
import numpy as np
from sklearn.datasets import load_iris, load_digits, load_breast_cancer, load_wi
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Gradien
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc


# Define datasets
datasets = {
    "Iris": load_iris(),
    "Digits": load_digits(),
    "Breast Cancer": load_breast_cancer(),
    "Wine": load_wine()
}


# Define parameter configurations for each classifier
model_params = {
    "Decision Tree": {"max_depth": [None, 3], "min_samples_split": [2, 5]},
    "Random Forest": {"n_estimators": [100, 200], "max_depth": [None, 5]},
    "AdaBoost": {"n_estimators": [50, 100], "learning_rate": [1.0, 0.5]},
    "Gradient Boost": {"n_estimators": [50, 100], "learning_rate": [0.1, 0.05]}
}


# Initialize a dictionary to store the results
results = {}

# Loop through each dataset
for dataset_name, dataset in datasets.items():
    print(f"Dataset: {dataset_name}")
    X = dataset.data
    y = dataset.target

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

    results[dataset_name] = {}

    # Loop through each classifier and its parameter configurations
    for model_name, params in model_params.items():
        print(f"Training {model_name}...")
        results[dataset_name][model_name] = {}

        for param_name, param_values in params.items():
            for param_value in param_values:
                # Create and train the classifier with the current parameter con
                if model_name == "Decision Tree":
                    classifier = DecisionTreeClassifier(**{param_name: param_valu
                elif model_name == "Random Forest":
                    classifier = RandomForestClassifier(**{param_name: param_valu
                elif model_name == "AdaBoost":
                    classifier = AdaBoostClassifier(**{param_name: param_value})
```

```
            classifier = AdaBoostClassifier(**{param_name: param_value})
        elif model_name == "Gradient Boost":
            classifier = GradientBoostingClassifier(**{param_name: param_
        else:
            raise ValueError(f"Unknown model name: {model_name}")

        classifier.fit(X_train, y_train)

        # Make predictions on the test set
        y_pred = classifier.predict(X_test)

        # Calculate evaluation metrics
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average="weighted")
        recall = recall_score(y_test, y_pred, average="weighted")
        f1 = f1_score(y_test, y_pred, average="weighted")

        # Store the results
        results[dataset_name][model_name][(param_name, param_value)] = {
            "Accuracy": accuracy,
            "Precision": precision,
            "Recall": recall,
            "F1-score": f1
        }
        print(f"{param_name}: {param_value}, Accuracy: {accuracy:.2f}, P:
```

```
Dataset: Iris
Training Decision Tree...
max_depth: None, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1
max_depth: 3, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1.00
min_samples_split: 2, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-sco
min_samples_split: 5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-sco
Training Random Forest...
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 200, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
max_depth: None, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1
max_depth: 5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score: 1.00
Training AdaBoost...
n_estimators: 50, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
learning_rate: 1.0, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
learning_rate: 0.5, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
Training Gradient Boost...
n_estimators: 50, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
n_estimators: 100, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score:
learning_rate: 0.1, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-score
learning_rate: 0.05, Accuracy: 1.00, Precision: 1.00, Recall: 1.00, F1-scor
Dataset: Digits
Training Decision Tree...
max_depth: None, Accuracy: 0.87, Precision: 0.87, Recall: 0.87, F1-score: 0
max_depth: 3, Accuracy: 0.38, Precision: 0.43, Recall: 0.38, F1-score: 0.32
min_samples_split: 2, Accuracy: 0.87, Precision: 0.87, Recall: 0.87, F1-sco
min_samples_split: 5, Accuracy: 0.85, Precision: 0.85, Recall: 0.85, F1-sco
Training Random Forest...
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, msg_start, len(result))
```

```
n_estimators: 100, Accuracy: 0.97, Precision: 0.97, Recall: 0.97, F1-score:
n_estimators: 200, Accuracy: 0.98, Precision: 0.98, Recall: 0.98, F1-score:
max_depth: None, Accuracy: 0.98, Precision: 0.98, Recall: 0.98, F1-score: 0
max_depth: 5, Accuracy: 0.95, Precision: 0.95, Recall: 0.95, F1-score: 0.95
Training AdaBoost...
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, msg_start, len(result))
n_estimators: 50, Accuracy: 0.34, Precision: 0.34, Recall: 0.34, F1-score:
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, msg_start, len(result))
n_estimators: 100, Accuracy: 0.34, Precision: 0.34, Recall: 0.34, F1-score:
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, msg_start, len(result))
learning_rate: 1.0, Accuracy: 0.34, Precision: 0.34, Recall: 0.34, F1-score
learning_rate: 0.5, Accuracy: 0.61, Precision: 0.74, Recall: 0.61, F1-score
Training Gradient Boost...
n_estimators: 50, Accuracy: 0.96, Precision: 0.97, Recall: 0.96, F1-score:
n_estimators: 100, Accuracy: 0.97, Precision: 0.97, Recall: 0.97, F1-score:
learning_rate: 0.1, Accuracy: 0.97, Precision: 0.97, Recall: 0.97, F1-score
learning_rate: 0.05, Accuracy: 0.96, Precision: 0.97, Recall: 0.96, F1-scor
Dataset: Breast Cancer
Training Decision Tree...
max_depth: None, Accuracy: 0.94, Precision: 0.94, Recall: 0.94, F1-score: 0
max_depth: 3, Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1-score: 0.96
min_samples_split: 2, Accuracy: 0.94, Precision: 0.95, Recall: 0.94, F1-sco
min_samples_split: 5, Accuracy: 0.95, Precision: 0.95, Recall: 0.95, F1-sco
Training Random Forest...
n_estimators: 100, Accuracy: 0.97, Precision: 0.97, Recall: 0.97, F1-score:
```

```
# Print the results
print("\nResults:")
for dataset_name, dataset_results in results.items():
    print(f"\nDataset: {dataset_name}")
    for model_name, params in dataset_results.items():
        print(f"\n{model_name}:")
        for param_config, metrics in params.items():
            print(f"{param_config}: {metrics}")
```

```
Results:

Dataset: Iris

Decision Tree:
('max_depth', None): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1
('max_depth', 3): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1-sc
('min_samples_split', 2): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0
('min_samples_split', 5): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0

Random Forest:
('n_estimators', 100): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('n_estimators', 200): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('max_depth', None): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1
('max_depth', 5): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F1-sc

AdaBoost:
('n_estimators', 50): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F
('n_estimators', 100): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('learning_rate', 1.0): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,
```

```
('learning_rate', 0.5): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,

Gradient Boost:
('n_estimators', 50): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, 'F
('n_estimators', 100): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0, '
('learning_rate', 0.1): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,
('learning_rate', 0.05): {'Accuracy': 1.0, 'Precision': 1.0, 'Recall': 1.0,

Dataset: Digits

Decision Tree:
('max_depth', None): {'Accuracy': 0.8711111111111111, 'Precision': 0.872167
('max_depth', 3): {'Accuracy': 0.38, 'Precision': 0.43033413848631236, 'Rec
('min_samples_split', 2): {'Accuracy': 0.8666666666666667, 'Precision': 0.8
('min_samples_split', 5): {'Accuracy': 0.8533333333333334, 'Precision': 0.8

Random Forest:
('n_estimators', 100): {'Accuracy': 0.9733333333333334, 'Precision': 0.9738
('n_estimators', 200): {'Accuracy': 0.9755555555555555, 'Precision': 0.9759
('max_depth', None): {'Accuracy': 0.9755555555555555, 'Precision': 0.976110
('max_depth', 5): {'Accuracy': 0.9466666666666667, 'Precision': 0.947710442

AdaBoost:
('n_estimators', 50): {'Accuracy': 0.33555555555555555, 'Precision': 0.3371
('n_estimators', 100): {'Accuracy': 0.33555555555555555, 'Precision': 0.337
('learning_rate', 1.0): {'Accuracy': 0.33555555555555555, 'Precision': 0.33
('learning_rate', 0.5): {'Accuracy': 0.6088888888888889, 'Precision': 0.737

Gradient Boost:
('n_estimators', 50): {'Accuracy': 0.9644444444444444, 'Precision': 0.96596
('n_estimators', 100): {'Accuracy': 0.9733333333333334, 'Precision': 0.9743
('learning_rate', 0.1): {'Accuracy': 0.9711111111111111, 'Precision': 0.972
('learning_rate', 0.05): {'Accuracy': 0.9644444444444444, 'Precision': 0.96

Dataset: Breast Cancer

Decision Tree:
```