Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

**Лабораторна робота №1**

з дисципліни «Технології машинного навчання»

Тема:

«Метрики якості задач класифікації»

Виконав студент групи ІА-12:
Бутрій Віталій

Перевірив:
Коломоєць С.О.

Київ 2024

**Мета роботи**: отримати знання основних метрик якості бінарної класифікації і варіантів тонкого налаштування алгоритмів класифікації.

**Завдання**

1) Завантажте файл bioresponse.csv з папки “Лаб 1” за посиланням https://drive.google.com/drive/folders/1SYLvcaJ5XZQL2socPuGc6466AoEsrLue?usp=sharing

```
dataset = pd.read_csv('bioresponse.csv')
x = dataset.iloc[:, 1:]
y = dataset.Activity
```

2) Навчіть 4 класифікатора, щоб передбачити поле "Activity" (біологічна відповідь молекули) з набору даних "bioresponse.csv":

- дрібне дерево рішень;
- глибоке дерево рішень;
- випадковий ліс на дрібних деревах;
- випадковий ліс на глибоких деревах.

Для імплементації кожного класифаєра створю базовий клас, який буде мати загальну структуру та функціональність:

```
class ClassifierBase:
    def __init__(self, model, model_name):
        self.model = model
        self.model_name = model_name

    def train(self, x_train, y_train):
        self.model.fit(x_train, y_train)

    def predict(self, x_test):
```

```
        return self.model.predict(x_test), self.model.predict_proba(x_test)[:,
1]

    def evaluate(self, x_test, y_test):
        y_pred, y_pred_prob = self.predict(x_test)

        print(f"=== {self.model_name} ===")
        print(f"Accuracy: {MetricsCalculator.get_accuracy(y_pred, y_test)}")
        print(f"Precision: {MetricsCalculator.get_precision(y_pred, y_test)}")
        print(f"Recall: {MetricsCalculator.get_recall(y_pred, y_test)}")
        print(f"F1 Score: {MetricsCalculator.get_f1_score(y_pred, y_test)}")
        print(f"Log Loss: {MetricsCalculator.get_log_loss(y_pred_prob,
y_test)}")

        Visualizer.show_roc_curve(self.model_name, y_pred_prob, y_test)
        Visualizer.show_precision_recall_curve(self.model_name, y_pred_prob,
y_test)
```

Клас Visualizer в свою чергу відповідає за побудову графіків precision-recall і ROC-кривих:

```
class Visualizer:
    @staticmethod
    def show_roc_curve(model_name, pred_values_prob, true_values):
        roc_auc = roc_auc_score(true_values, pred_values_prob)
        false_positive_rates, true_positive_rates, _ = roc_curve(true_values,
pred_values_prob)
        plt.plot([0, 1], [0, 1], '--')
        plt.plot(false_positive_rates, true_positive_rates, label=f'ROC AUC =
{roc_auc:.2f}')
        plt.xlabel("False positive rate")
        plt.ylabel("True positive rate")
        plt.title(f"{model_name} ROC Curve")
        plt.legend()
        plt.show()

    @staticmethod
    def show_precision_recall_curve(model_name, pred_values_prob,
true_values):
```

```
        precision, recall, thresholds = precision_recall_curve(true_values,
pred_values_prob)
        plt.plot(thresholds, precision[:-1], label='Precision', color='blue')
        plt.plot(thresholds, recall[:-1], label='Recall', color='orange')
        plt.xlabel("Threshold")
        plt.ylabel("Score")
        plt.title(f"{model_name} Precision-Recall Curve")
        plt.legend()
        plt.show()
```

В результаті, кожен класифікатор буде наслідником базового (очевидно, з вказанням своїх особливих параметрів в конструкторі):

```
class ShallowTreeClassifier(ClassifierBase):
    def __init__(self):
        from sklearn.tree import DecisionTreeClassifier
        super().__init__(DecisionTreeClassifier(max_depth=5, random_state=25),
"Shallow Tree")


class DeepTreeClassifier(ClassifierBase):
    def __init__(self):
        from sklearn.tree import DecisionTreeClassifier
        super().__init__(DecisionTreeClassifier(max_depth=20,
random_state=25), "Deep Tree")


class ShallowForestClassifier(ClassifierBase):
    def __init__(self):
        from sklearn.ensemble import RandomForestClassifier
        super().__init__(RandomForestClassifier(n_estimators=100, max_depth=5,
random_state=25), "Shallow Forest")


class DeepForestClassifier(ClassifierBase):
    def __init__(self):
        from sklearn.ensemble import RandomForestClassifier
        super().__init__(RandomForestClassifier(n_estimators=100,
max_depth=20, random_state=25), "Deep Forest")
```

```
class AvoidTypeIIErrorClassifier(ClassifierBase):
    def __init__(self):
        from sklearn.tree import DecisionTreeClassifier
        super().__init__(DecisionTreeClassifier(max_depth=5, class_weight={0:
1, 1: 5}, random_state=25),
                         "Avoid Type II Error Tree")
```

3) Розрахуйте наступні метрики, щоб перевірити якість ваших моделей:

- частка правильних відповідей (accuracy);
- точність;
- повнота;
- *F*1-score
- log-loss.

Для метрик виділю також окремий клас:

```
class MetricsCalculator:
    @staticmethod
    def get_accuracy(pred_values, true_values):
        correct_predictions = (pred_values == true_values).sum()
        return correct_predictions / len(true_values)

    @staticmethod
    def get_precision(pred_values, true_values):
        true_positives = ((pred_values == 1) & (true_values == 1)).sum()
        false_positives = ((pred_values == 1) & (true_values == 0)).sum()
        return true_positives / (true_positives + false_positives)

    @staticmethod
    def get_recall(pred_values, true_values):
        true_positives = ((pred_values == 1) & (true_values == 1)).sum()
        false_negatives = ((pred_values == 0) & (true_values == 1)).sum()
        return true_positives / (true_positives + false_negatives)

    @staticmethod
```

```
    def get_f1_score(pred_values, true_values):
        precision = MetricsCalculator.get_precision(pred_values, true_values)
        recall = MetricsCalculator.get_recall(pred_values, true_values)
        return 2 * precision * recall / (precision + recall)

    @staticmethod
    def get_log_loss(pred_values_prob, true_values):
        log_loss = -((true_values * np.log(pred_values_prob + 1e-8)) +
                     (1 - true_values) * np.log(1 - pred_values_prob +
1e-8)).mean()
        return log_loss
```

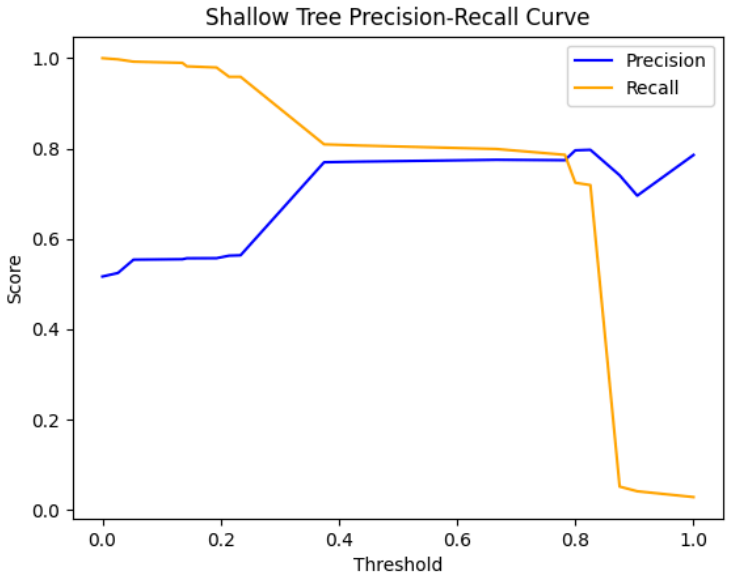
Його функції використовуються у базовому класі класифікатора, що дозволяє мені обчислити метрики для кожної моделі.

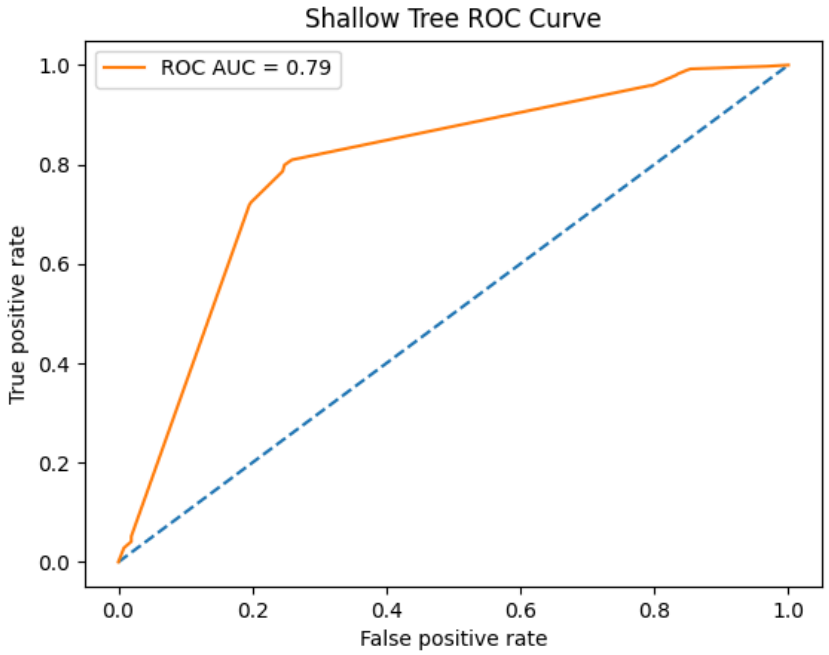
Після інтерпретації скрипту lab1.py маємо такі результати метрик:

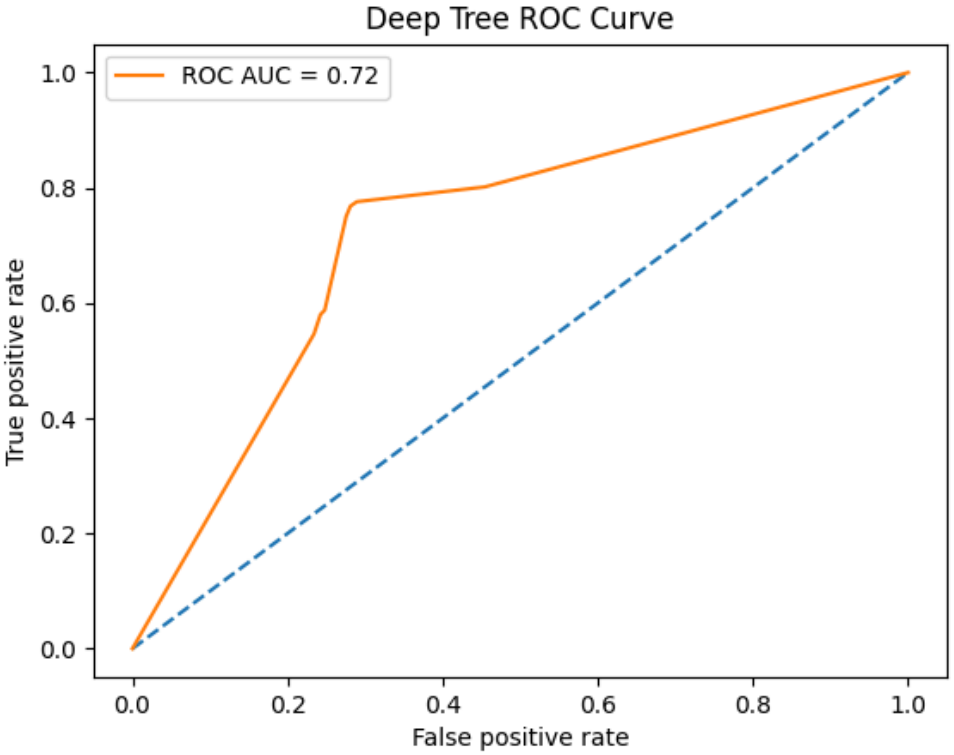
```
/Users/vb/PycharmProjects/TMN/.venv/bin/python /Users/vb/PycharmProjects/TMN/lab1.py
=== Shallow Tree ===
Accuracy: 0.7762982689747004
Precision: 0.775
Recall: 0.7989690721649485
F1 Score: 0.7868020304568527
Log Loss: 0.6132956230192237
=== Deep Tree ===
Accuracy: 0.7443408788282291
Precision: 0.7437810945273632
Recall: 0.770618556701031
F1 Score: 0.7569620253164557
Log Loss: 4.107685754539268
=== Shallow Forest ===
Accuracy: 0.7323568575233023
Precision: 0.7391304347826086
Recall: 0.7448453608247423
F1 Score: 0.741976893453145
Log Loss: 0.5617016737185694
=== Deep Forest ===
Accuracy: 0.8149134487350199
Precision: 0.8029197080291971
Recall: 0.8505154639175257
F1 Score: 0.8260325406758448
Log Loss: 0.4563112965956851
=== Avoid Type II Error Tree ===
Accuracy: 0.6231691078561917
Precision: 0.5826771653543307
Recall: 0.9536082474226805
F1 Score: 0.7233626588465298
Log Loss: 0.8754448647653811

Process finished with exit code 0
```

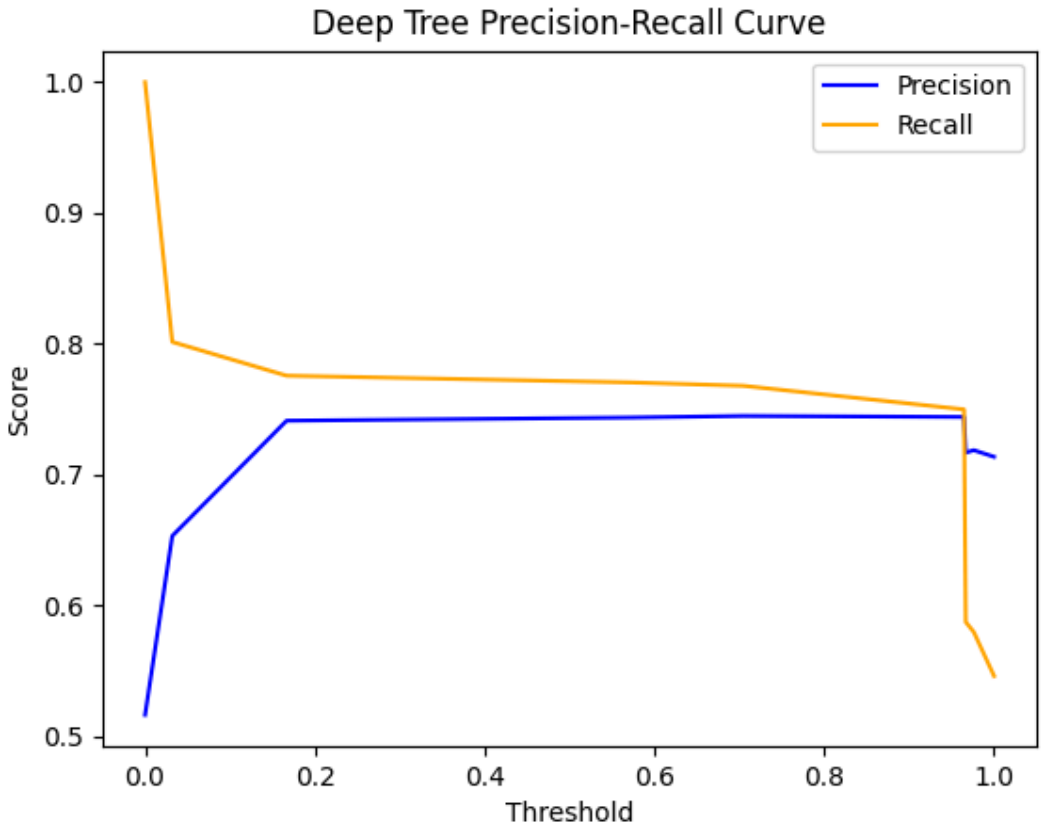
4) Побудуйте precision-recall і ROC-криві для ваших моделей.

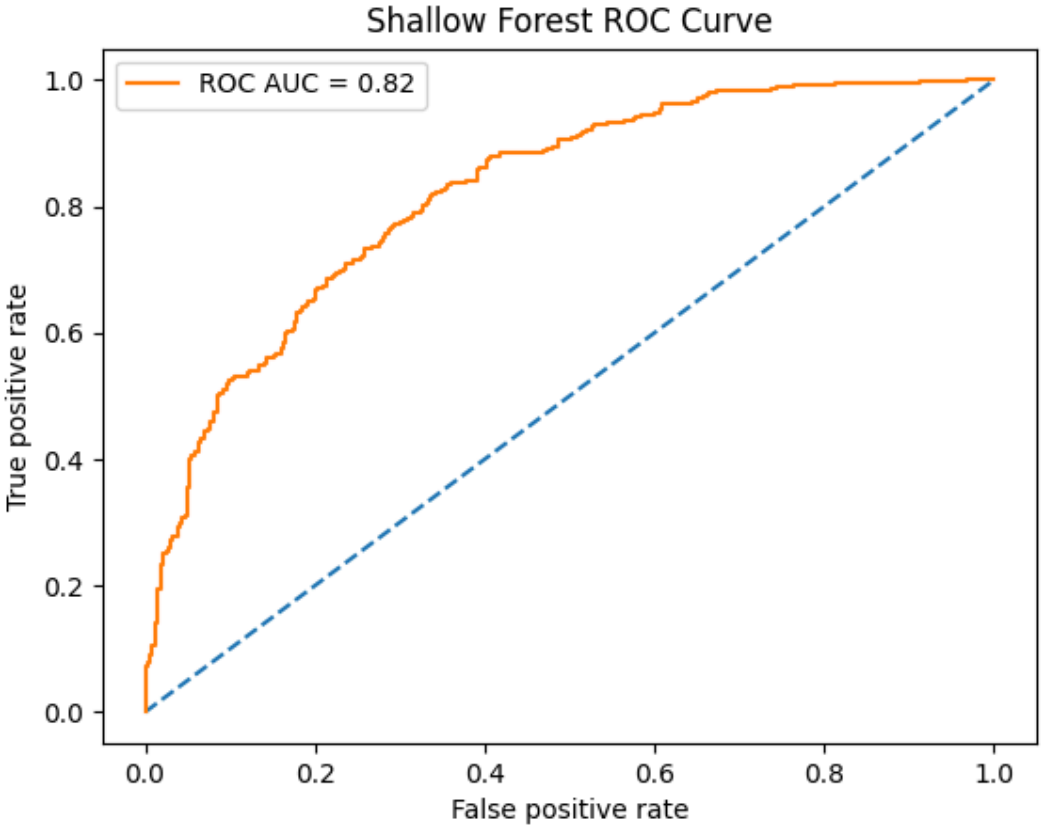
Маємо по 2 графіки для кожної моделі:

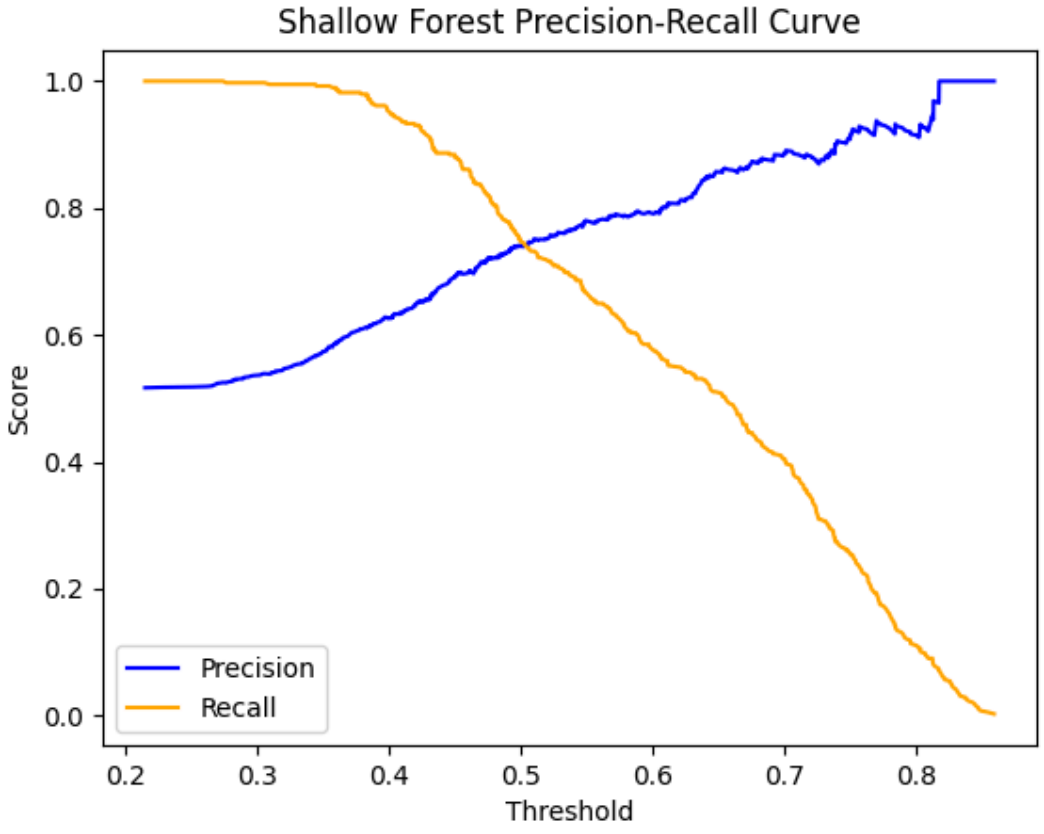

Shallow Tree Precision-Recall Curve
Precision
Recall
Score
Threshold
0.0
0.2
0.4
0.6
0.8
1.0

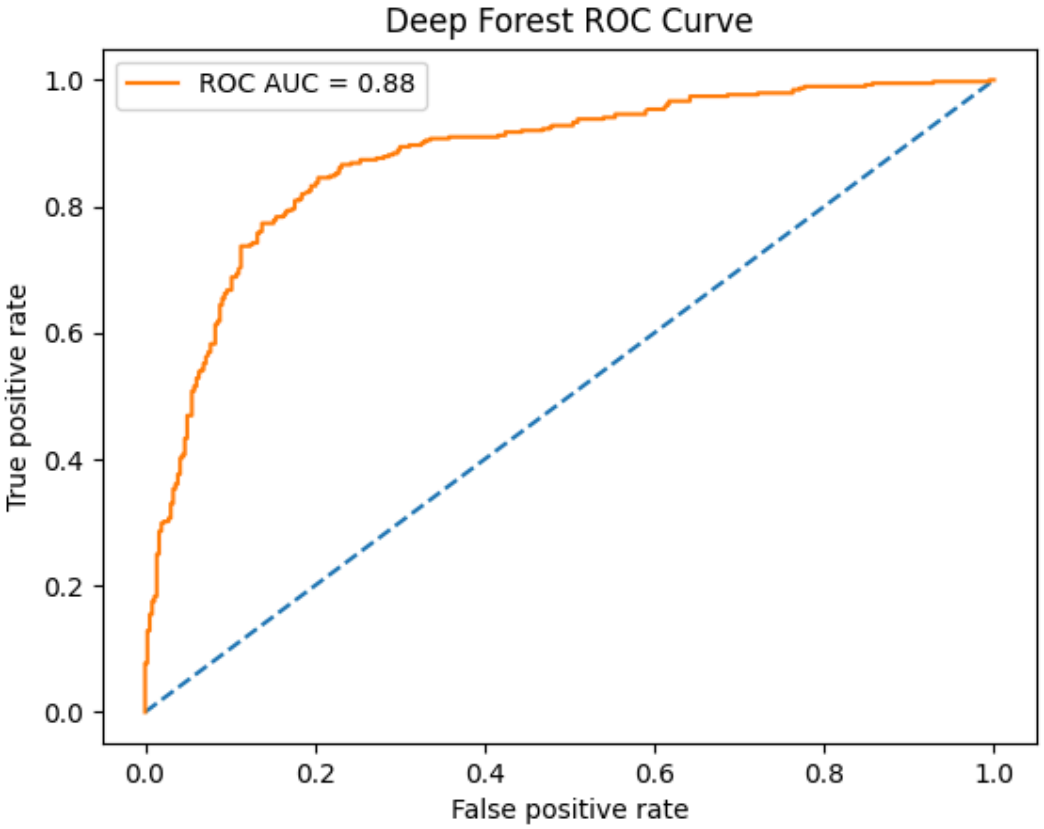

Shallow Tree ROC Curve
ROC AUC = 0.79
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

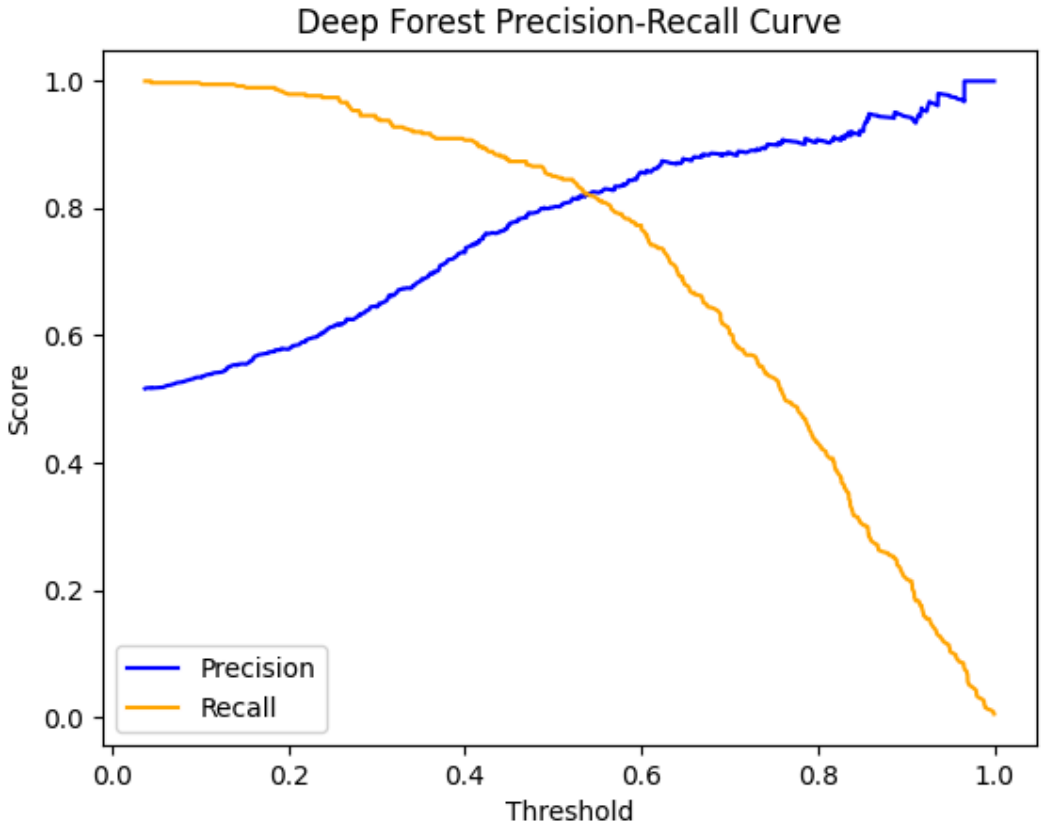

Deep Tree ROC Curve
ROC AUC = 0.72
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

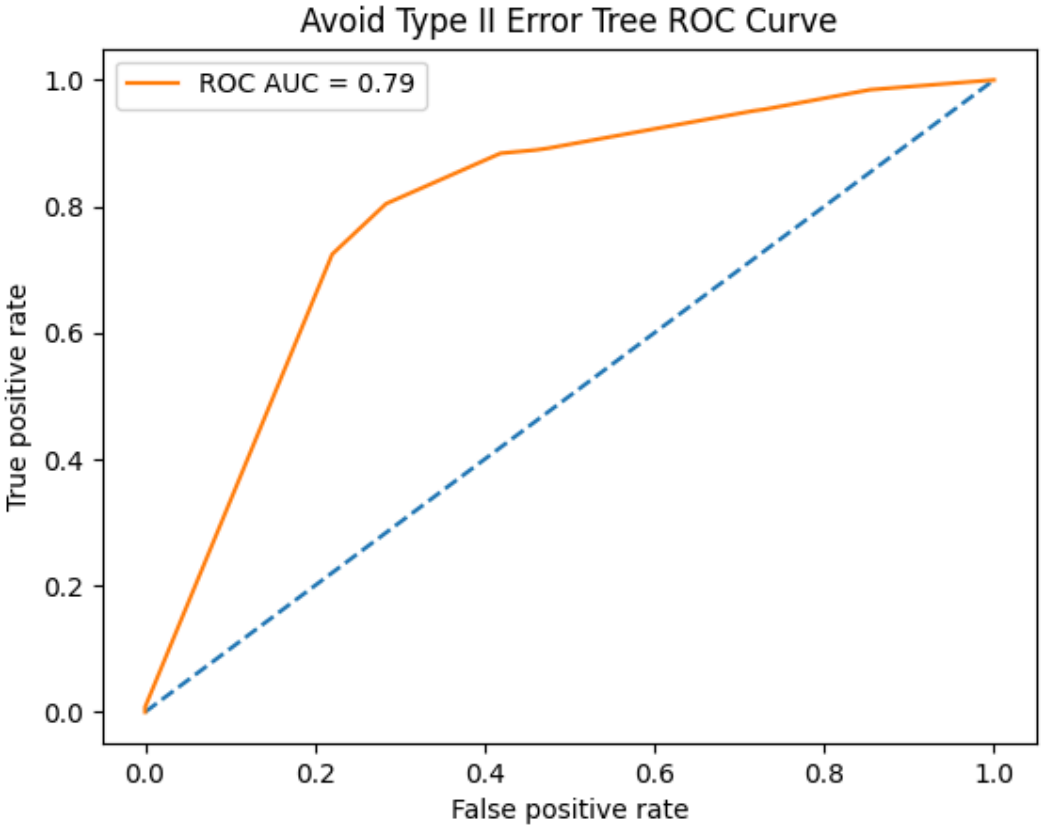

Deep Tree Precision-Recall Curve
Precision
Recall
Score
Threshold
0.5
0.6
0.7
0.8
0.9
1.0
0.0
0.2
0.4
0.6
0.8
1.0

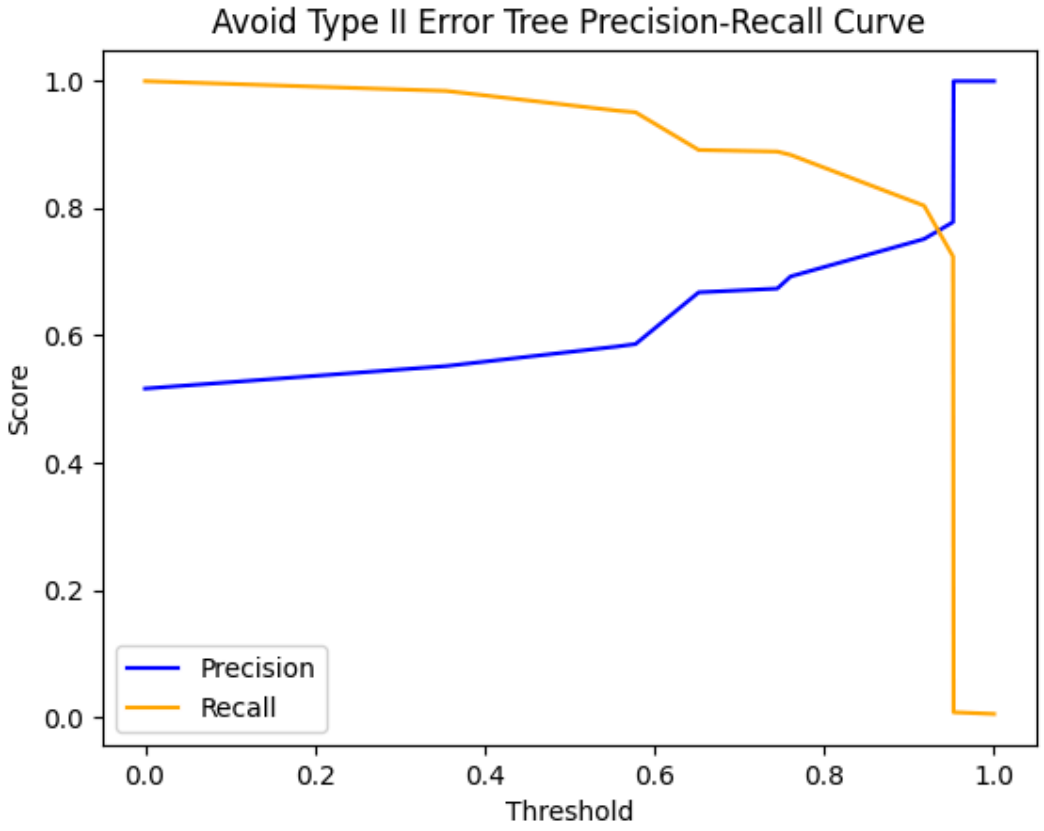

Shallow Forest ROC Curve
ROC AUC = 0.82
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


Shallow Forest Precision-Recall Curve
Precision
Recall
Score
Threshold
0.0
0.2
0.4
0.6
0.8
1.0
0.2
0.3
0.4
0.5
0.6
0.7
0.8

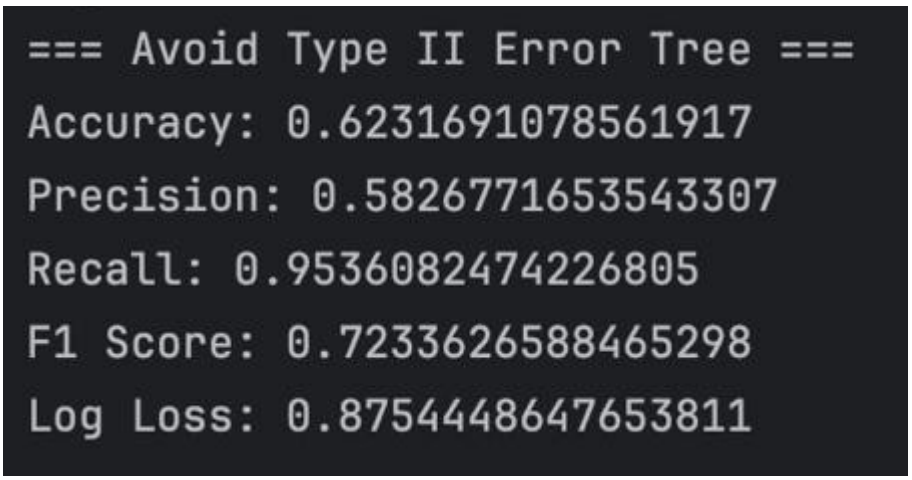

Deep Forest ROC Curve
ROC AUC = 0.88
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


Deep Forest Precision-Recall Curve
Precision
Recall
Score
Threshold
0.0
0.2
0.4
0.6
0.8
1.0


Avoid Type II Error Tree ROC Curve
ROC AUC = 0.79
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


Avoid Type II Error Tree Precision-Recall Curve
Precision
Recall
Score
Threshold
0.0
0.2
0.4
0.6
0.8
1.0

5) Навчіть класифікатор, який уникає помилок II роду і розрахуйте для нього метрики якості.




Ця модель налаштована на уникнення Type II помилок (пропуск позитивів), жертвуючи точністю. Підходить для критичних задач, де важливий високий recall (наприклад, медичні діагнози).

**Висновок:**

У цій лабораторній роботі було реалізовано та досліджено кілька ключових функцій, що забезпечують тренування, оцінювання та візуалізацію результатів для моделей машинного навчання, зокрема дерева рішень та випадкового лісу. Моделі навчаються на тренувальних даних і перевіряються за допомогою таких метрик, як точність, precision, recall, F1-міра та log loss. Для кожної моделі створюються ROC-криві та криві Precision-Recall, які дають змогу оцінити її ефективність у задачах класифікації. Крім того, код ілюструє можливість адаптації моделі для зменшення помилок другого роду (типу II) шляхом зміни ваг класів у дереві рішень.