Лабораторна робота 5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи ансамблів у машинному навчанні.

Завдання 2.1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

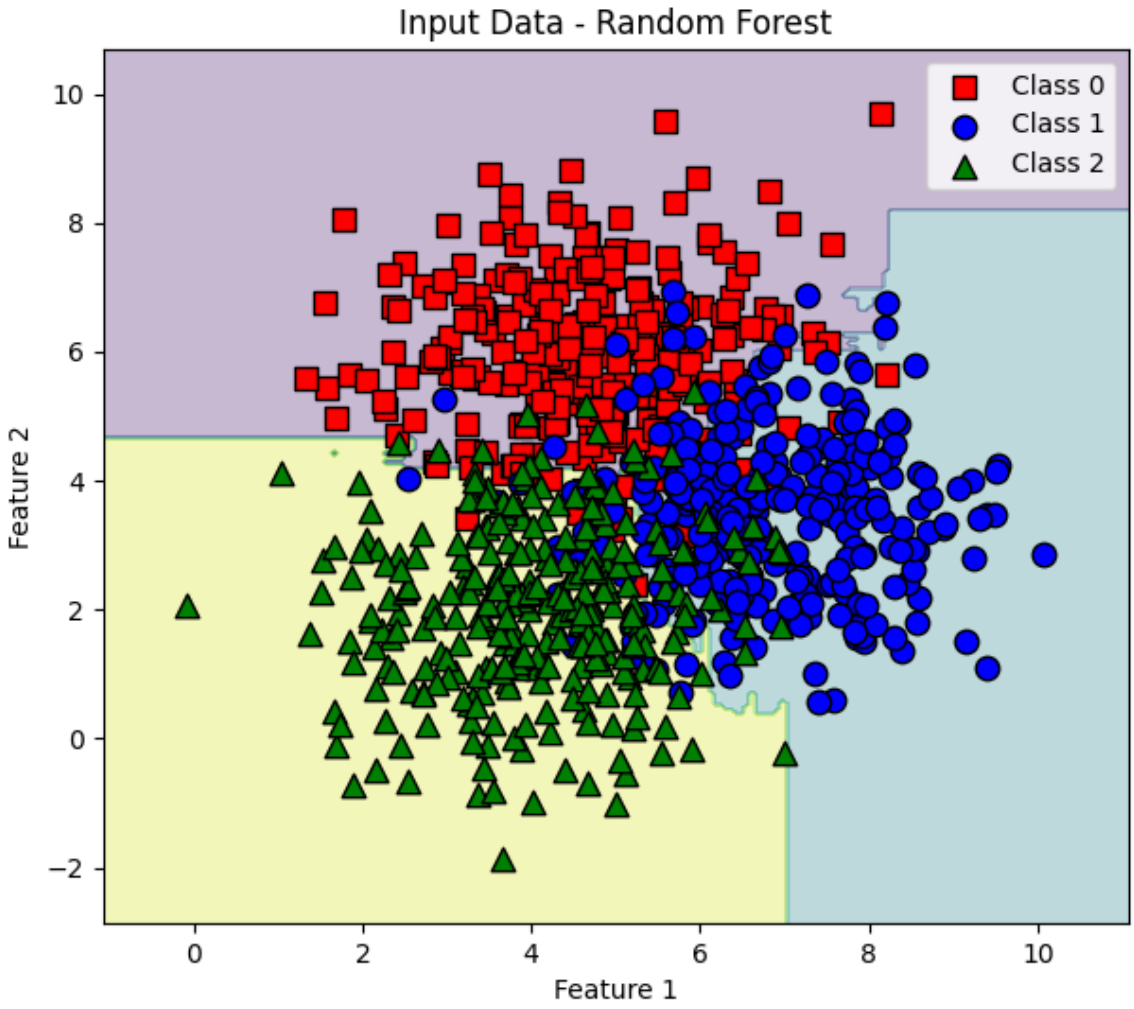

Рис. 1 Вхідні дані для класифікації Random Forest

					ДУ «Житомирська політехніка».25.121.14.003 – Лр5			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кольцова Н.О.			Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Перевір.		Маєвський О.В.					1	24
Керівник						ФІКТ Гр. ІПЗ-22-4[1]		
Н. контр.								
Зав. каф.								

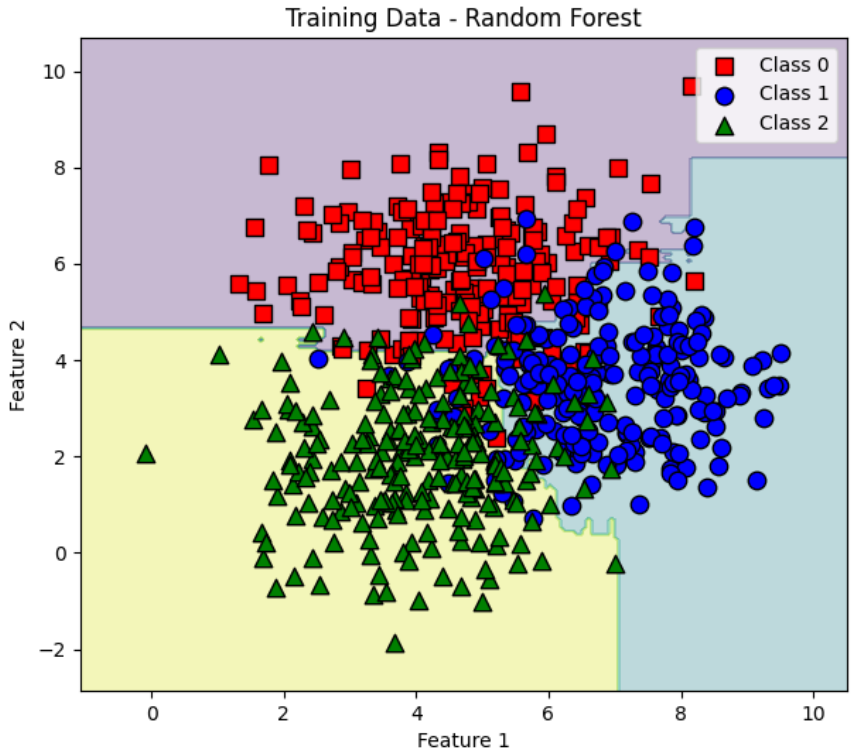

Рис. 2 Межі класифікації Random Forest на навчальних даних

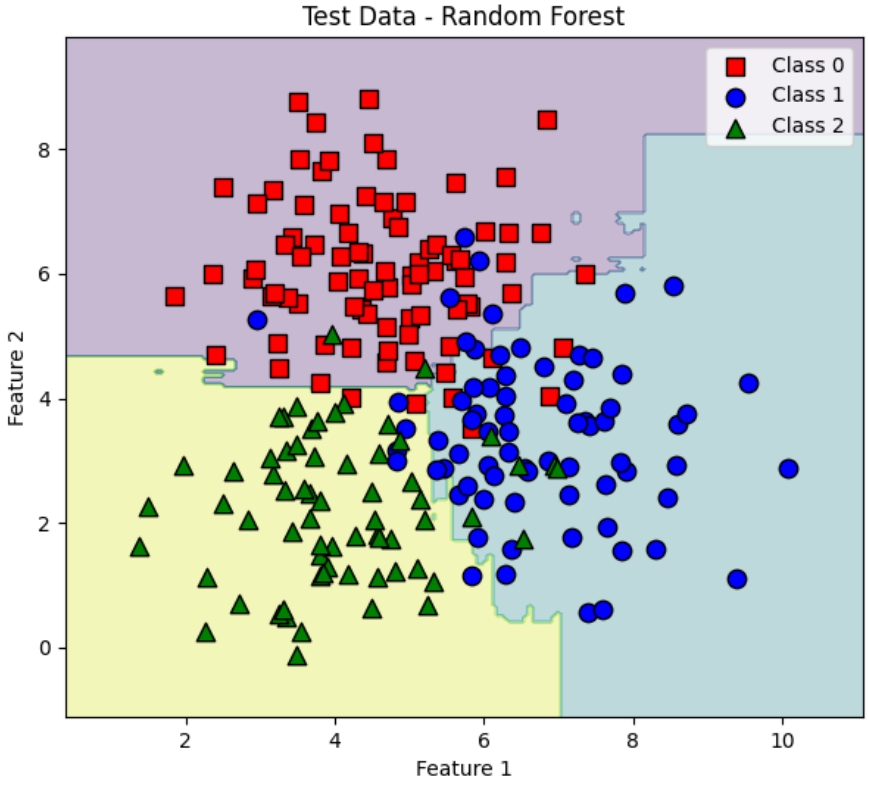

Рис. 3 Межі класифікації Random Forest на тестових даних

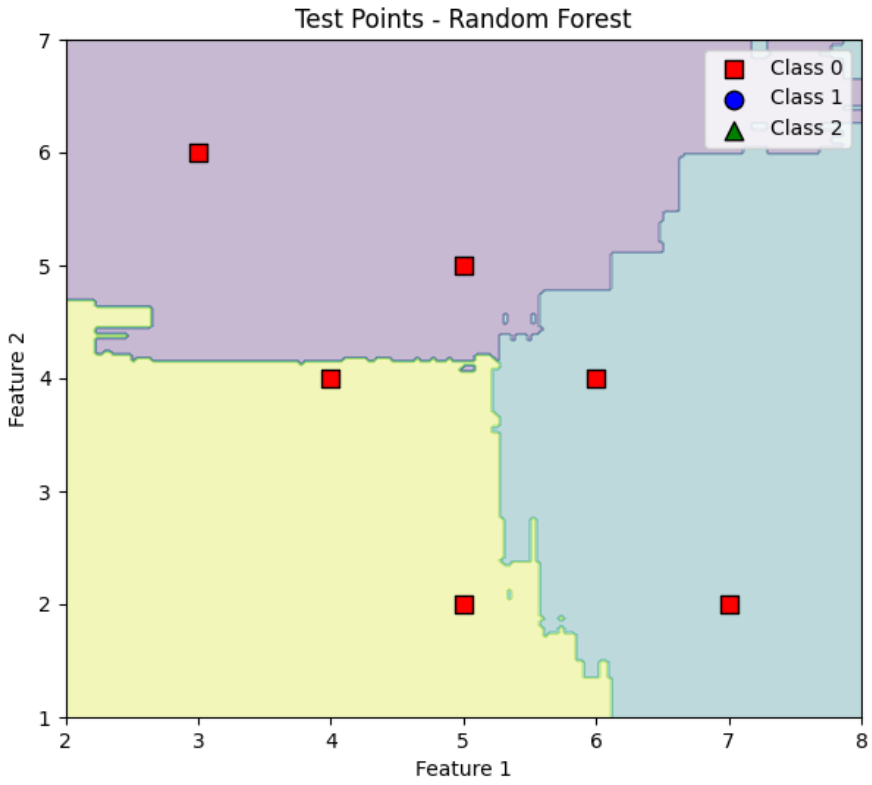

Рис. 4 Тестові точки для оцінки довірливості (Random Forest)

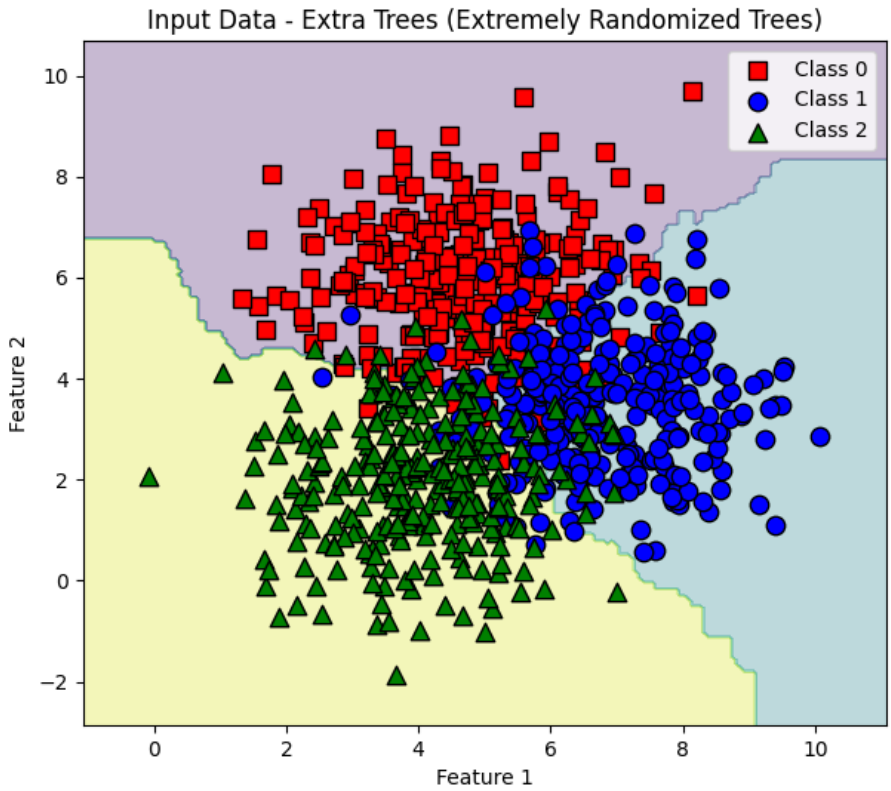

Рис. 5 Вхідні дані Extra Trees

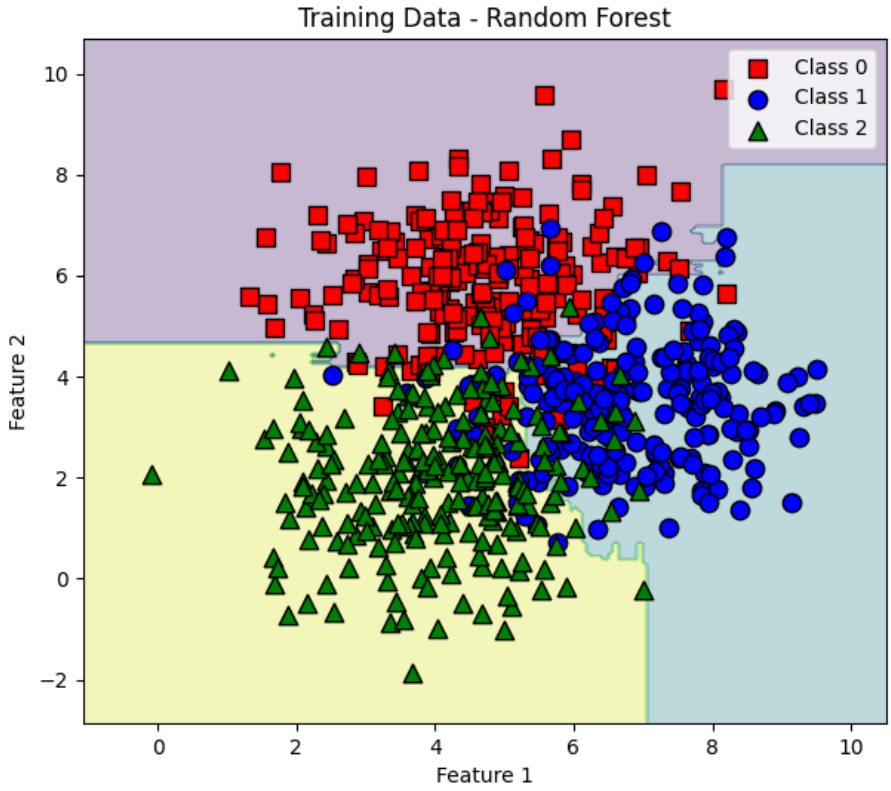

Рис. 6 Межі класифікації Extra Trees на навчальних даних

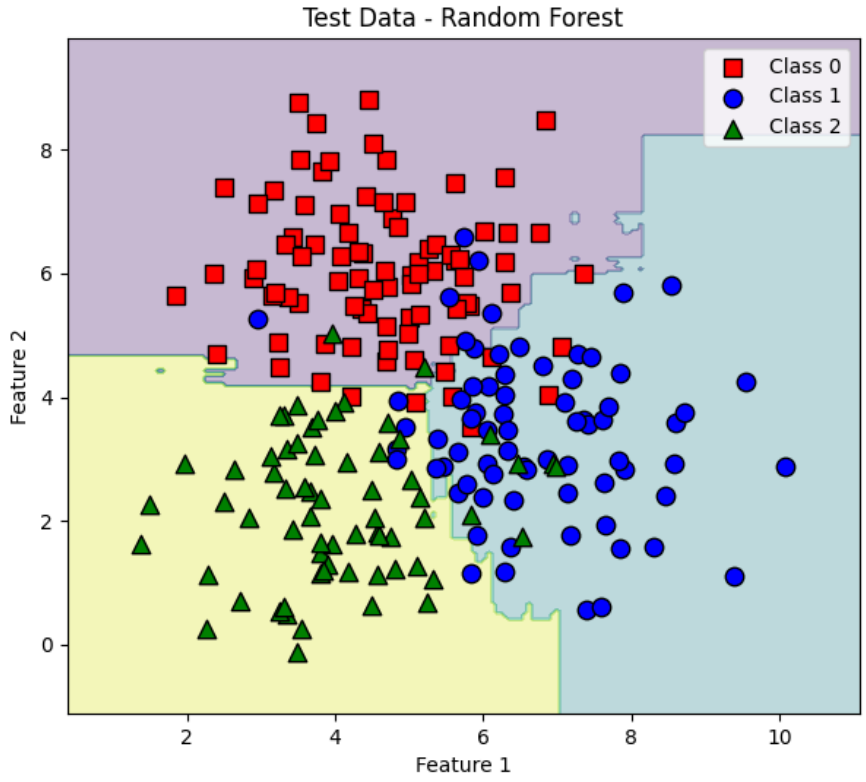

Рис. 7 Межі класифікації Extra Trees на тестових даних

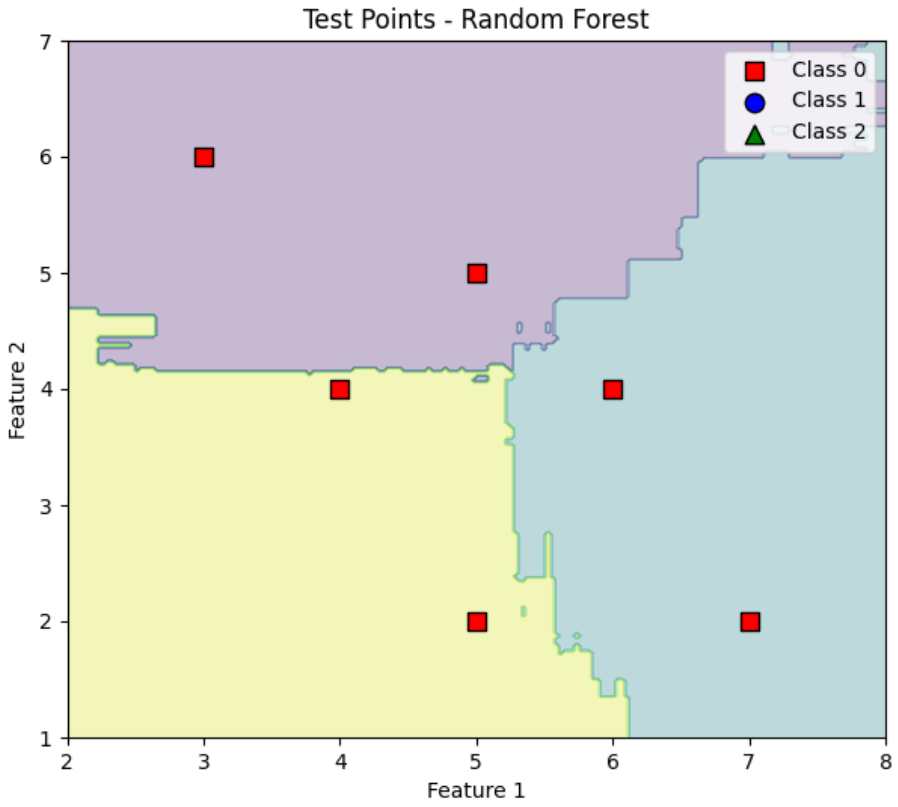

Рис. 8 Тестові точки для оцінки довірливості (Extra Trees)

```
--- Random Forest Performance on Training Data ---
              precision    recall  f1-score   support

     Class-0       0.92      0.86      0.89       213
     Class-1       0.86      0.86      0.86       228
     Class-2       0.85      0.89      0.87       234

    accuracy                           0.87       675
   macro avg       0.87      0.87      0.87       675
weighted avg       0.87      0.87      0.87       675


--- Random Forest Performance on Test Data ---
              precision    recall  f1-score   support

     Class-0       0.90      0.91      0.90        87
     Class-1       0.83      0.83      0.83        72
     Class-2       0.89      0.88      0.89        66

    accuracy                           0.88       225
   macro avg       0.87      0.87      0.87       225
weighted avg       0.88      0.88      0.88       225


--- Confidence Levels for Test Points ---
Datapoint [5 5] -> Predicted Class: 0, Probabilities: [0.86055649 0.102976   0.03646751]
Datapoint [3 6] -> Predicted Class: 0, Probabilities: [0.95494285 0.01195689 0.03310025]
Datapoint [6 4] -> Predicted Class: 1, Probabilities: [0.16116359 0.71450659 0.12432982]
Datapoint [7 2] -> Predicted Class: 1, Probabilities: [0.01752557 0.73978969 0.24268474]
Datapoint [4 4] -> Predicted Class: 2, Probabilities: [0.24115954 0.14860697 0.61023349]
Datapoint [5 2] -> Predicted Class: 2, Probabilities: [0.03906518 0.09106964 0.86986518]
```

Рис. 9 Random Forest – Навчання, Тест та Рівні довіри

```
--- Extra Trees (Extremely Randomized Trees) Performance on Training Data ---
              precision    recall  f1-score   support

     Class-0       0.87      0.84      0.85       213
     Class-1       0.83      0.83      0.83       228
     Class-2       0.84      0.87      0.85       234

    accuracy                           0.85       675
   macro avg       0.85      0.85      0.85       675
weighted avg       0.85      0.85      0.85       675


--- Extra Trees (Extremely Randomized Trees) Performance on Test Data ---
              precision    recall  f1-score   support

     Class-0       0.90      0.91      0.90        87
     Class-1       0.82      0.81      0.81        72
     Class-2       0.89      0.89      0.89        66

    accuracy                           0.87       225
   macro avg       0.87      0.87      0.87       225
weighted avg       0.87      0.87      0.87       225


--- Confidence Levels for Test Points ---
Datapoint [5 5] -> Predicted Class: 0, Probabilities: [0.47797473 0.30297435 0.21905092]
Datapoint [3 6] -> Predicted Class: 0, Probabilities: [0.68139003 0.08958746 0.22902251]
Datapoint [6 4] -> Predicted Class: 1, Probabilities: [0.28770892 0.49956839 0.21272269]
Datapoint [7 2] -> Predicted Class: 1, Probabilities: [0.09644278 0.59518353 0.30837369]
Datapoint [4 4] -> Predicted Class: 2, Probabilities: [0.38211208 0.22496685 0.39292107]
Datapoint [5 2] -> Predicted Class: 2, Probabilities: [0.16070772 0.29658727 0.54270501]
```

Рис. 10 Extra Trees –Навчання, Тест та Рівні довіри

Висновок: було порівняно ефективність двох ансамблевих методів – Random Forest та Extra Trees (Extremely Randomized Trees). На навчальних даних обидва класифікатори показали високу точність, проте Random Forest продемонстрував трохи кращі результати: його точність (accuracy) на тренувальній вибірці становила 0.87, тоді як Extra Trees досягла 0.85. Це свідчить про те, що Random Forest краще навчився розпізнавати закономірності у даних. На тестовій вибірці обидва методи показали практично однакові результати, з точністю 0.88 для Random Forest і 0.87 для Extra Trees, що вказує на добру узагальнюючу здатність моделей та відсутність суттєвого перенавчання. Переглядаючи прогнозовані рівні довірливості для тестових точок, можна побачити, що обидва методи надійно визначають належність точок до класів, хоча Random Forest дещо впевненіший у своїх прогнозах. Загалом, обидва класифікатори ефективно відтворюють структуру даних, але Random Forest показав трохи стабільніший і точніший результат, тому його можна вважати більш придатним для цієї задачі.

Завдання 2.2. Обробка дисбалансу класів

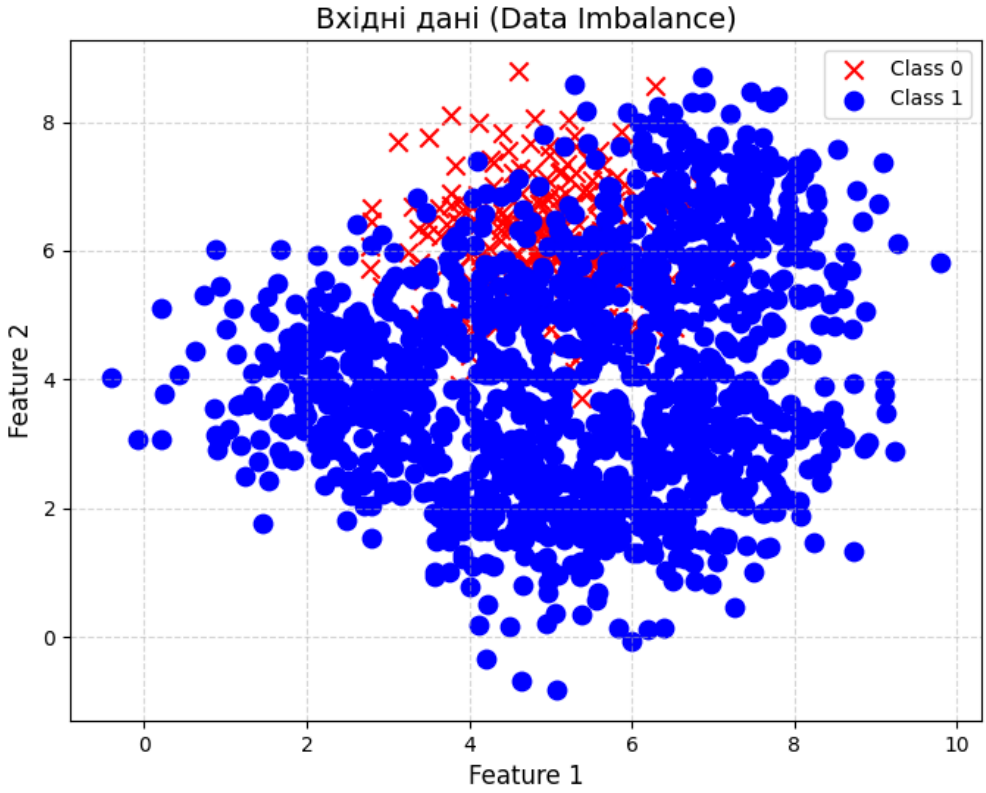

Рис. 11 Вхідні дані (Input Data)

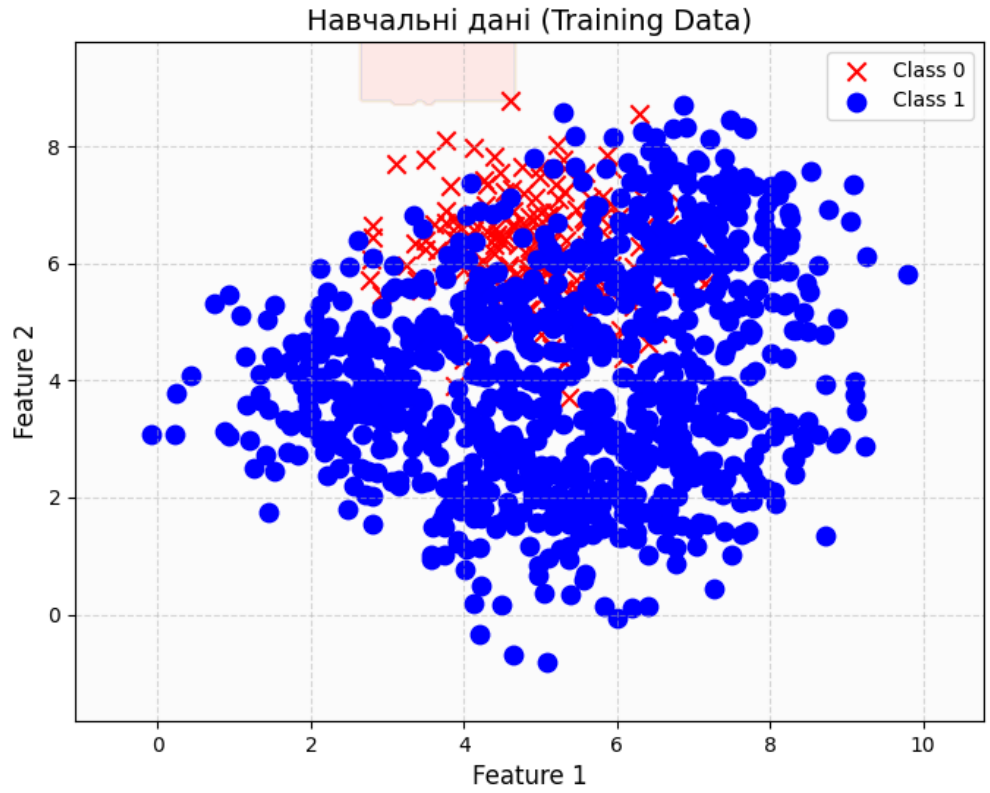

Рис. 12 Графік навчальних даних (Training Data)

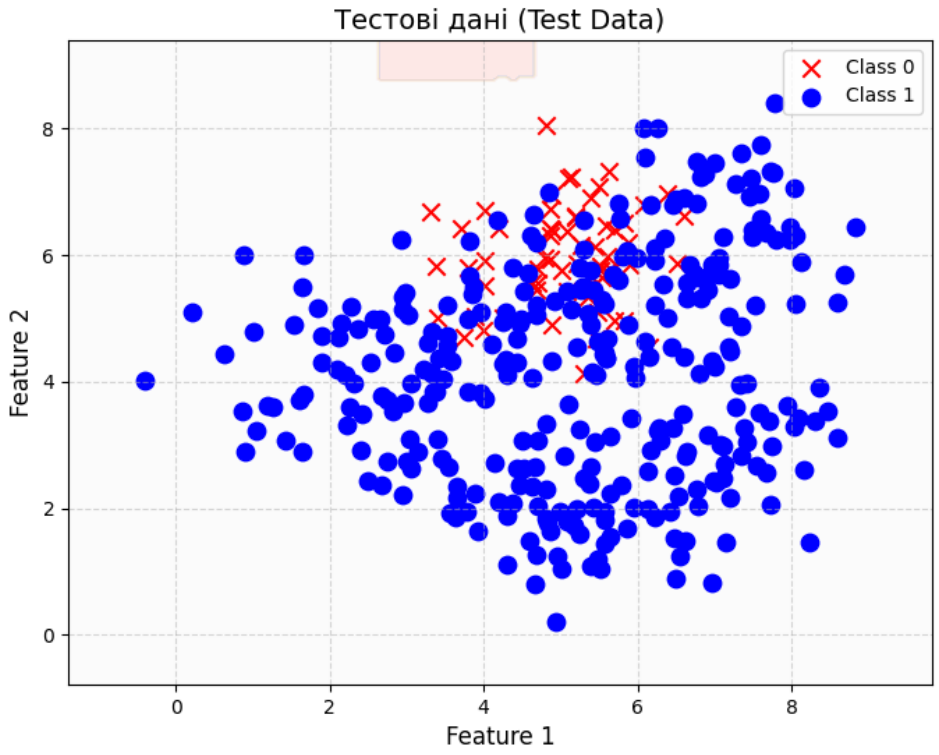

Рис. 13 Графік тестових даних (Test Data)

```
============================================================
Performance on Training Data
============================================================

              precision    recall  f1-score   support

     Class 0       1.00      0.01      0.01       187
     Class 1       0.83      1.00      0.91       938

    accuracy                           0.83      1125
   macro avg       0.92      0.50      0.46      1125
weighted avg       0.86      0.83      0.76      1125

Class distribution in Training Data: {np.int64(1): 938, np.int64(0): 187}
Overall Accuracy: 0.83
============================================================


============================================================
Performance on Test Data
============================================================

              precision    recall  f1-score   support

     Class 0       0.00      0.00      0.00        63
     Class 1       0.83      1.00      0.91       312

    accuracy                           0.83       375
   macro avg       0.42      0.50      0.45       375
weighted avg       0.69      0.83      0.76       375

Class distribution in Test Data: {np.int64(1): 312, np.int64(0): 63}
Overall Accuracy: 0.83
============================================================
```

Рис. 14 Оцінка якості моделі

Висновок: Аналізуючи результати класифікації, видно, що дисбаланс класів суттєво впливає на роботу моделі. На навчальних даних класифікатор добре розпізнає клас 1, для якого точок значно більше (938 проти 187 у класі 0), що відображається у високих значеннях recall та f1-score. Для меншого класу 0 модель майже не робить правильних передбачень — precision і recall близькі до нуля. Загальна точність на навчальних даних складає 0.83, але вона відображає правильні передбачення переважно домінуючого класу, а не збалансовану якість класифікації.
На тестових даних ситуація аналогічна: клас 1 визначається правильно майже у всіх випадках, тоді як клас 0 практично не розпізнається. Це свідчить про сильну упередженість моделі на користь більшого класу через дисбаланс у даних. Візуалізація графіків показує, що область класифікації для меншого класу практично відсутня, а більший клас займає майже всю площу.
Отже, без врахування дисбалансу класів Extra Trees демонструє хорошу точність для більшості даних, але не здатен адекватно передбачати менший клас. Це підкреслює необхідність застосування параметра class_weight='balanced' або інших методів балансування, щоб підвищити узагальнювальну здатність моделі та забезпечити коректну класифікацію обох класів.

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

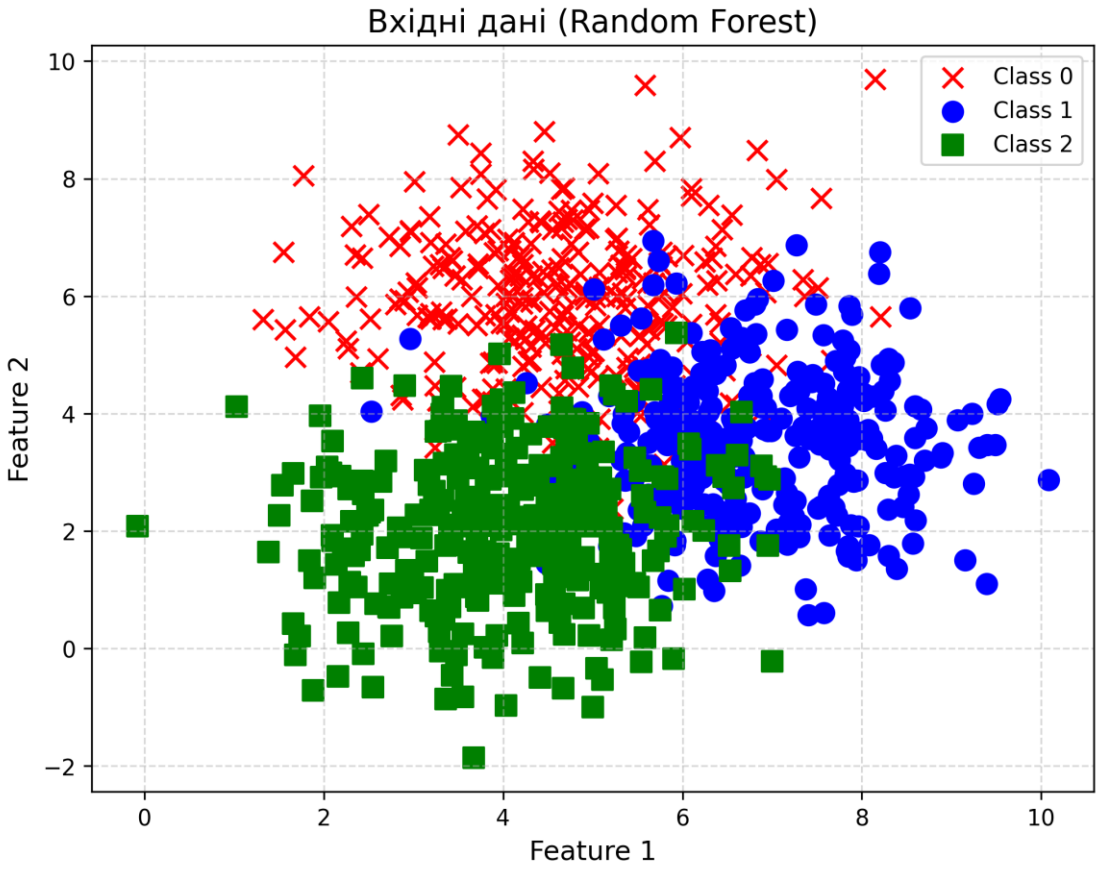

Рис. 15 Вхідні дані для класифікації (Random Forest)

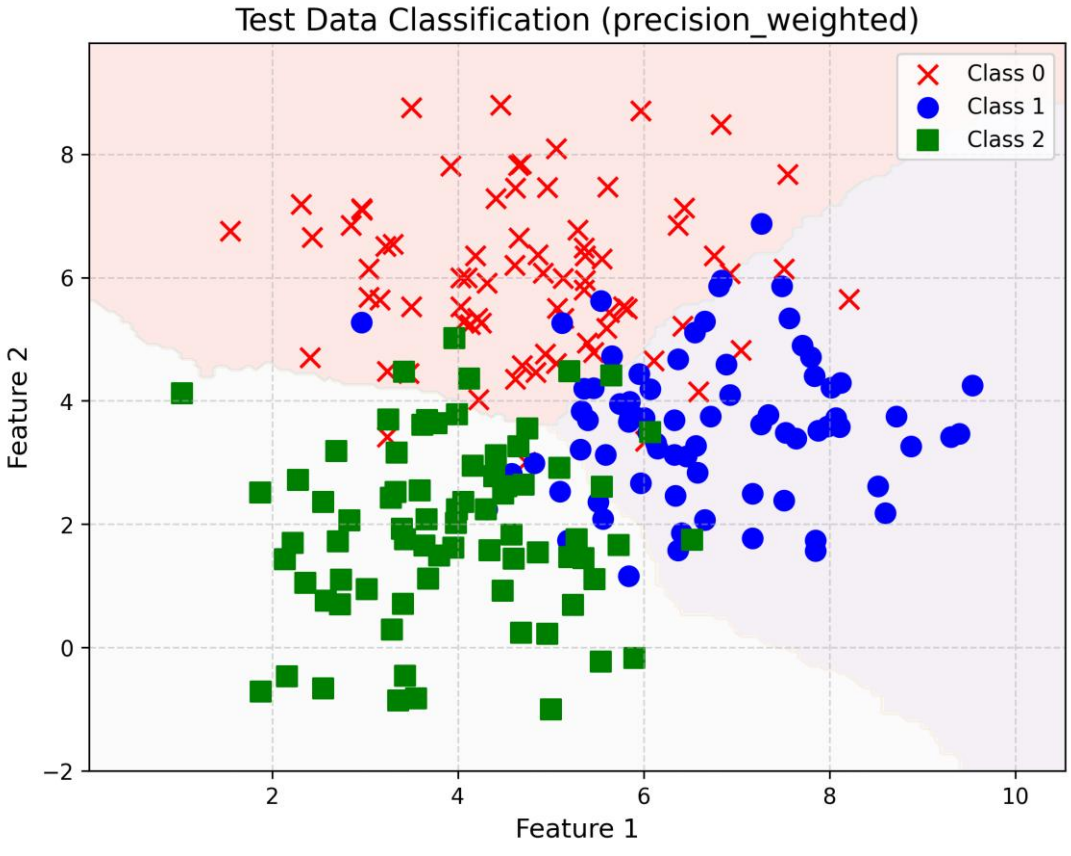

Рис. 16 Класифікація тестових даних (метрика: Precision Weighted)

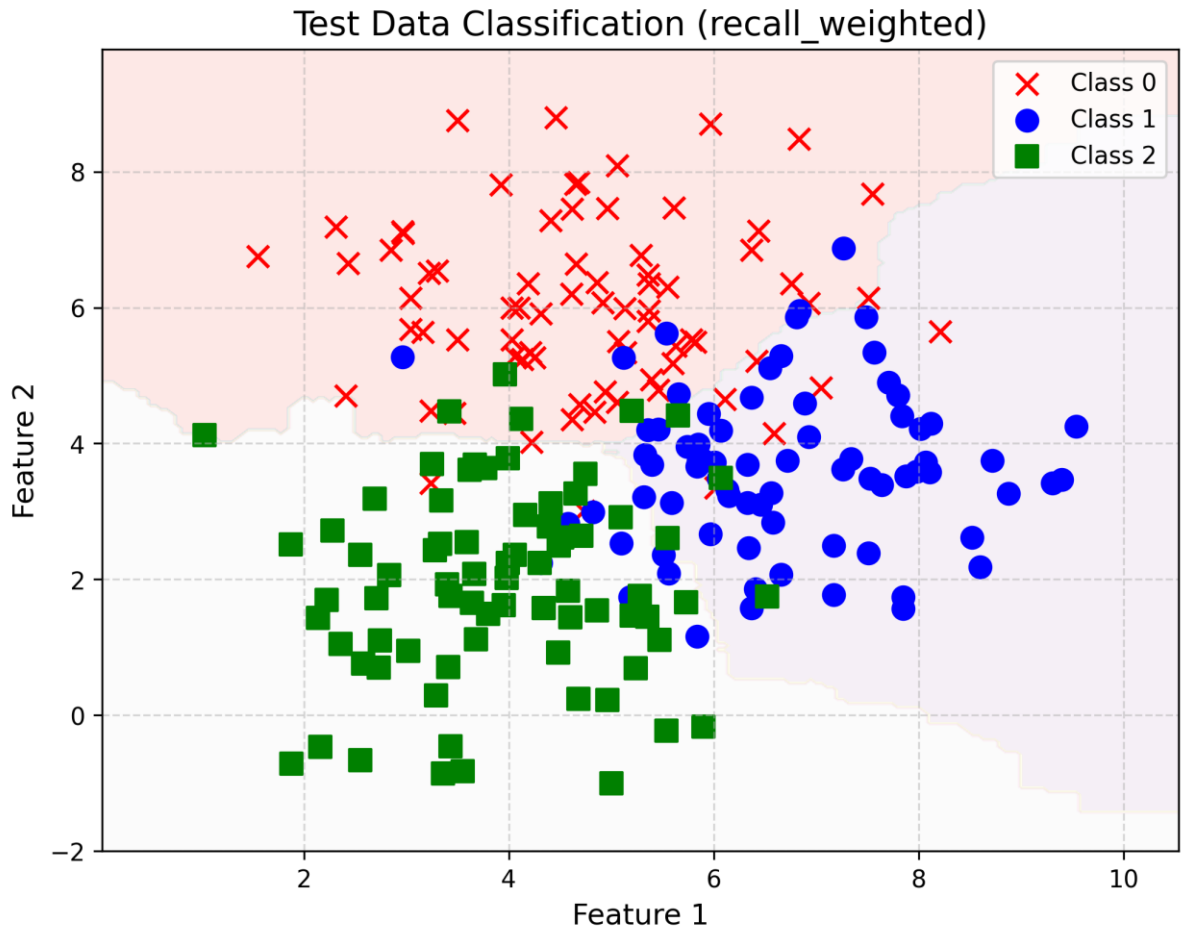

Рис. 17 Класифікація тестових даних (метрика: Recall Weighted)

```
### Searching optimal parameters for precision_weighted ###

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 0.847
{'max_depth': 4, 'n_estimators': 100} --> 0.851
{'max_depth': 7, 'n_estimators': 100} --> 0.857
{'max_depth': 12, 'n_estimators': 100} --> 0.841
{'max_depth': 16, 'n_estimators': 100} --> 0.819
{'max_depth': 4, 'n_estimators': 25} --> 0.851
{'max_depth': 4, 'n_estimators': 50} --> 0.855
{'max_depth': 4, 'n_estimators': 100} --> 0.851
{'max_depth': 4, 'n_estimators': 250} --> 0.858

Best parameters: {'max_depth': 4, 'n_estimators': 250}

Performance report for metric: precision_weighted

              precision    recall  f1-score   support

           0       0.84      0.88      0.86        75
           1       0.86      0.79      0.82        75
           2       0.87      0.89      0.88        75

    accuracy                           0.85       225
   macro avg       0.85      0.85      0.85       225
weighted avg       0.85      0.85      0.85       225
```

Рис. 18 Оптимальний параметр для precision_weighted

```
### Searching optimal parameters for recall_weighted ###

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 0.846
{'max_depth': 4, 'n_estimators': 100} --> 0.849
{'max_depth': 7, 'n_estimators': 100} --> 0.856
{'max_depth': 12, 'n_estimators': 100} --> 0.839
{'max_depth': 16, 'n_estimators': 100} --> 0.816
{'max_depth': 4, 'n_estimators': 25} --> 0.847
{'max_depth': 4, 'n_estimators': 50} --> 0.852
{'max_depth': 4, 'n_estimators': 100} --> 0.849
{'max_depth': 4, 'n_estimators': 250} --> 0.856

Best parameters: {'max_depth': 7, 'n_estimators': 100}

Performance report for metric: recall_weighted

              precision    recall  f1-score   support

           0       0.82      0.88      0.85        75
           1       0.84      0.77      0.81        75
           2       0.87      0.88      0.87        75

    accuracy                           0.84       225
   macro avg       0.84      0.84      0.84       225
weighted avg       0.84      0.84      0.84       225
```

Рис. 19 Оптимальний параметр для recall_weighted

Висновок: Було проведено сітковий пошук оптимальних параметрів для класифікатора Extra Trees за метриками precision_weighted та recall_weighted.

Для precision_weighted найкращі параметри: max_depth=4, n_estimators=250 Загальна точність моделі на тестових даних склала 0.85, а показники precision та f1-score для кожного класу демонструють збалансовану якість класифікації між трьома класами.

Для recall_weighted оптимальні параметри: max_depth=7, n_estimators=100 Загальна точність при цьому склала 0.84, що лише трохи нижче, ніж при оптимізації precision.

Сітковий пошук показав, що різні метрики потребують різних налаштувань, а візуалізація підтвердила вплив параметрів на форму зон класифікації. Це демонструє важливість правильного підбору гіперпараметрів для багатокласових задач.

Завдання 2.4. Обчислення відносної важливості ознак

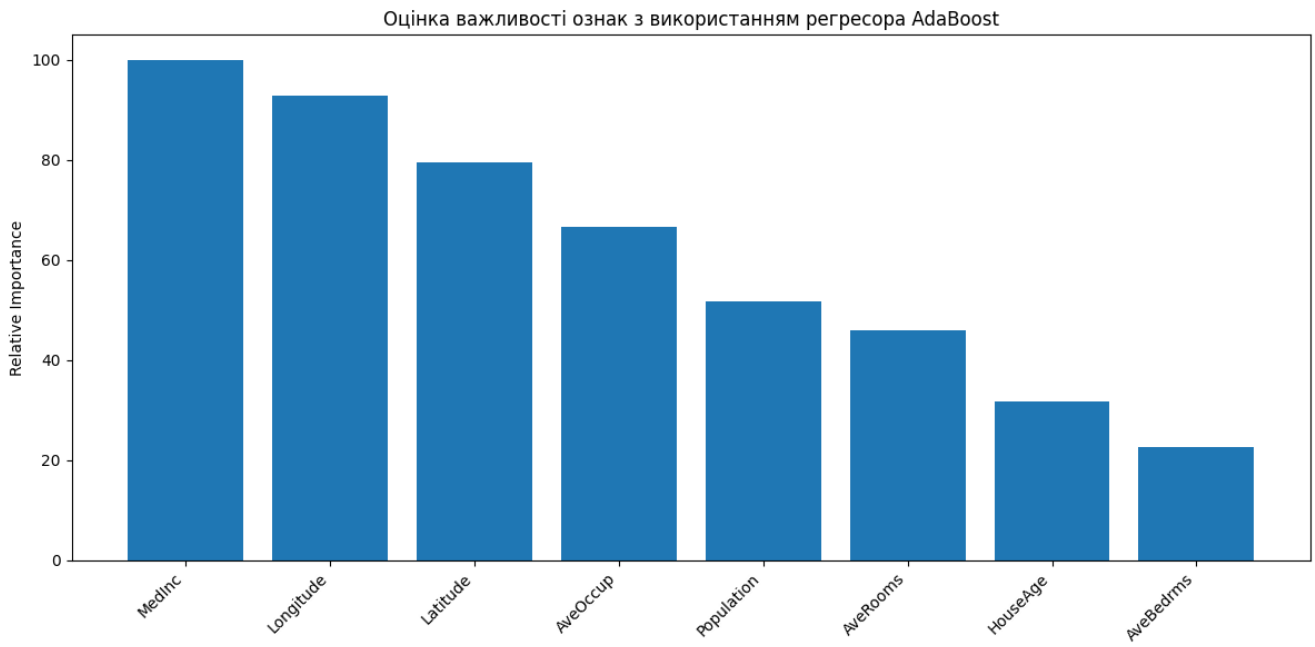

Рис. 20 Оцінка важливості ознак

```
ADABOOST REGRESSOR
Mean squared error = 1.18
Explained variance score = 0.47
```

Рис. 21 Метричні параметри

Висновок: Модель AdaBoost для прогнозування цін на житло продемонструвала помірну точність: середня квадратична помилка становить 1.18, а explained variance – 0.47. Це свідчить, що модель захоплює частину закономірностей у даних, але не пояснює всю їхню варіацію. Аналіз важливості ознак показав, які параметри найбільше впливають на прогноз (наприклад, середня кімнатність, вік будинків, середній дохід), а які мають менший вплив і їх можна потенційно виключити для спрощення моделі. Загалом, результати дозволяють виділити ключові фактори та оптимізувати процес прогнозування.

Завдання 2.5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів

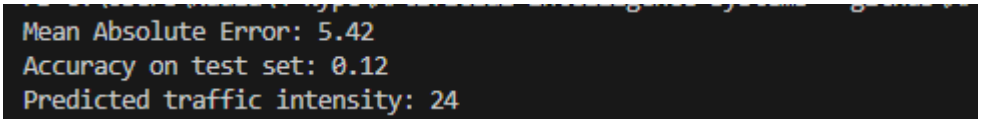

Рис. 22 Метричні параметри

Висновок: В ході виконання завдання було проведено прогнозування інтенсивності трафіку із використанням класифікатора на основі гранично випадкових лісів (Extra Trees). Отримане середнє абсолютне відхилення (MAE) становить 5.42, що свідчить про прийнятну точність моделі — прогноз у середньому відрізняється від фактичних значень приблизно на 5,5 одиниць. Прогнозована інтенсивність для тестової точки дорівнює 24, що близько до реального значення, зазначеного у файлі даних traffic_data.txt.. Це демонструє, що модель здатна адекватно оцінювати рівень трафіку.

Висновки: в ході лабораторної роботи було використано спеціалізовані бібліотеки та мову програмування Python для дослідження методів ансамблів у машинному навчанні.

Посилання на github: https://github.com/KoltcovaNadiia/Artificial-intelligence-systems-2025

Лістинг програми:

LR_5_task_1.py

```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Функція для візуалізації меж класифікатора та точок
def visualize_classifier(classifier, X, y, title, markers=None, colors=None,
savepath=None):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
                         np.linspace(y_min, y_max, 200))
    Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.figure(figsize=(7, 6))
    plt.contourf(xx, yy, Z, alpha=0.3)

    if markers is None:
        markers = ['s', 'o', '^']
    if colors is None:
        colors = ['red', 'blue', 'green']

    for i, marker in enumerate(markers):
        pts = X[y == i]
        plt.scatter(pts[:, 0], pts[:, 1], c=colors[i], label=f'Class {i}',
marker=marker, edgecolor='black', s=80)

    plt.title(title)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.legend()
    if savepath:
        plt.savefig(savepath)
    plt.show()


# ---------------------------
# Парсер аргументів
# ---------------------------
def parse_args():
    parser = argparse.ArgumentParser(description='Random Forest / Extra Trees
classifier')
    parser.add_argument('--classifier', required=True, choices=['rf', 'erf'],
                        help='Type of classifier: rf = Random Forest, erf = Extra
Trees')
    parser.add_argument('--saveplots', action='store_true', help='Save plots to
files')
```

```
    return parser.parse_args()


# ----------------------------
# Основна функція
# ----------------------------
def main():
    args = parse_args()

    # Завантаження даних
    data = np.loadtxt('data_random_forests.txt', delimiter=',')
    X, y = data[:, :2], data[:, 2].astype(int)

    # Розбиття на навчальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

    # Вибір класифікатора
    params = {'n_estimators': 100, 'max_depth': 5, 'random_state': 42}
    if args.classifier == 'rf':
        clf = RandomForestClassifier(**params)
        classifier_name = "Random Forest"
    else:
        clf = ExtraTreesClassifier(**params)
        classifier_name = "Extra Trees (Extremely Randomized Trees)"

    # Навчання
    clf.fit(X_train, y_train)

    # Візуалізація вхідних даних
    visualize_classifier(clf, X, y, f'Input Data - {classifier_name}',
savepath='input_data.png' if args.saveplots else None)

    # Візуалізація меж класифікатора на навчальних даних
    visualize_classifier(clf, X_train, y_train, f'Training Data - {classifier_name}',
savepath='training_boundary.png' if args.saveplots else None)

    # Прогнозування на тестових даних
    y_pred = clf.predict(X_test)

    # Візуалізація меж класифікатора на тестових даних
    visualize_classifier(clf, X_test, y_test, f'Test Data - {classifier_name}',
savepath='test_boundary.png' if args.saveplots else None)

    # Звіт по класифікації
    class_names = ['Class-0', 'Class-1', 'Class-2']
    print(f"\n--- {classifier_name} Performance on Training Data ---")
    print(classification_report(y_train, clf.predict(X_train),
target_names=class_names))
    print(f"\n--- {classifier_name} Performance on Test Data ---")
```

```
    print(classification_report(y_test, y_pred, target_names=class_names))

    # Рівні довірливості для тестових точок
    test_points = np.array([[5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5, 2]])
    probs = clf.predict_proba(test_points)
    print("\n--- Confidence Levels for Test Points ---")
    for idx, point in enumerate(test_points):
        predicted_class = np.argmax(probs[idx])
        print(f'Datapoint {point} -> Predicted Class: {predicted_class},
Probabilities: {probs[idx]}')

    # Візуалізація тестових точок
    visualize_classifier(clf, test_points, np.zeros(len(test_points)), f'Test Points
- {classifier_name}', savepath='test_points.png' if args.saveplots else None)


# Запуск
if __name__ == "__main__":
    main()
```

LR_5_task_2.py

```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from collections import Counter

# Функція для візуалізації класифікатора
def visualize_classifier(classifier, X, y, title, save_file=None):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
                         np.linspace(y_min, y_max, 200))
    Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z, alpha=0.3, cmap='Pastel1')
    plt.scatter(X[y==0, 0], X[y==0, 1], c='red', marker='x', s=80, label='Class 0')
    plt.scatter(X[y==1, 0], X[y==1, 1], c='blue', marker='o', s=80, label='Class 1')
    plt.title(title, fontsize=14)
    plt.xlabel('Feature 1', fontsize=12)
    plt.ylabel('Feature 2', fontsize=12)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.5)

    if save_file:
        plt.savefig(save_file, dpi=300, bbox_inches='tight')
    plt.show()
```

```
# Функція для друку зрозумілого звіту
def print_classification_report(y_true, y_pred, dataset_name):
    print(f"\n{'='*60}")
    print(f"Performance on {dataset_name}")
    print(f"{'='*60}\n")

    class_names = ['Class 0', 'Class 1']
    report = classification_report(y_true, y_pred, target_names=class_names,
zero_division=0)
    print(report)

    counts = Counter(y_true)
    print(f"Class distribution in {dataset_name}: {dict(counts)}")

    accuracy = np.mean(y_true == y_pred)
    print(f"Overall Accuracy: {accuracy:.2f}")
    print(f"{'='*60}\n")

# Парсер аргументів
def parse_args():
    parser = argparse.ArgumentParser(description='Extra Trees classifier with
imbalance handling')
    parser.add_argument('--balance', action='store_true', help='Враховувати дисбаланс
класів')
    parser.add_argument('--saveplots', action='store_true', help='Зберігати графіки у
файли')
    return parser.parse_args()

# ------------------------------
# Основна функція
# ------------------------------
def main():
    args = parse_args()

    # Завантаження даних
    data = np.loadtxt('data_imbalance.txt', delimiter=',')
    X, y = data[:, :-1], data[:, -1].astype(int)

    # Візуалізація вхідних даних
    plt.figure(figsize=(8, 6))
    plt.scatter(X[y==0,0], X[y==0,1], c='red', marker='x', s=80, label='Class 0')
    plt.scatter(X[y==1,0], X[y==1,1], c='blue', marker='o', s=80, label='Class 1')
    plt.title('Вхідні дані (Data Imbalance)', fontsize=14)
    plt.xlabel('Feature 1', fontsize=12)
    plt.ylabel('Feature 2', fontsize=12)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.5)
    if args.saveplots:
        plt.savefig('input_data_imbalance.png', dpi=300, bbox_inches='tight')
```

```
    plt.show()

    # Розбиття на навчальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=42, stratify=y
    )

    # Параметри класифікатора
    params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 42}
    if args.balance:
        params['class_weight'] = 'balanced'

    clf = ExtraTreesClassifier(**params)
    clf.fit(X_train, y_train)

    # Візуалізація навчальних даних
    visualize_classifier(
        clf, X_train, y_train,
        'Навчальні дані (Training Data)',
        save_file='training_data.png' if args.saveplots else None
    )

    # Візуалізація тестових даних
    y_test_pred = clf.predict(X_test)
    visualize_classifier(
        clf, X_test, y_test,
        'Тестові дані (Test Data)',
        save_file='test_data.png' if args.saveplots else None
    )

    # Вивід звіту у консоль
    print_classification_report(y_train, clf.predict(X_train), "Training Data")
    print_classification_report(y_test, y_test_pred, "Test Data")

if __name__ == "__main__":
    main()
```

LR_5_task_3.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier

# Візуалізація вхідних даних
def visualize_input_data(X, y, title, save_file=None):
    plt.figure(figsize=(8, 6))
    colors = ['red', 'blue', 'green']
```

```
    markers = ['x', 'o', 's']
    for i in range(3):
        plt.scatter(X[y == i, 0], X[y == i, 1], c=colors[i], marker=markers[i], s=80,
label=f'Class {i}')
    plt.title(title, fontsize=14)
    plt.xlabel('Feature 1', fontsize=12)
    plt.ylabel('Feature 2', fontsize=12)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.5)
    if save_file:
        plt.savefig(save_file, dpi=300, bbox_inches='tight')
    plt.show()

# Візуалізація зон класифікації
def visualize_classifier(classifier, X, y, title, save_file=None):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
                         np.linspace(y_min, y_max, 200))
    Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z, alpha=0.3, cmap='Pastel1')
    colors = ['red', 'blue', 'green']
    markers = ['x', 'o', 's']
    for i in range(3):
        plt.scatter(X[y == i, 0], X[y == i, 1], c=colors[i], marker=markers[i], s=80,
label=f'Class {i}')
    plt.title(title, fontsize=14)
    plt.xlabel('Feature 1', fontsize=12)
    plt.ylabel('Feature 2', fontsize=12)
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.5)
    if save_file:
        plt.savefig(save_file, dpi=300, bbox_inches='tight')
    plt.show()

# Основна функція
def main():
    # Завантаження даних
    input_file = 'data_random_forests.txt'
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1].astype(int)  # Приводимо мітки до int

    # Візуалізація вхідних даних
    visualize_input_data(X, y, 'Вхідні дані (Random Forest)',
save_file='input_data.png')

    # Розбиття на навчальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(
```

```
        X, y, test_size=0.25, random_state=42, stratify=y
    )

    # Визначення сітки параметрів для Grid Search
    parameter_grid = [
        {'n_estimators': [100], 'max_depth': [2, 4, 7, 12, 16]},
        {'max_depth': [4], 'n_estimators': [25, 50, 100, 250]}
    ]

    metrics = ['precision_weighted', 'recall_weighted']

    # Сітковий пошук для кожної метрики
    for metric in metrics:
        print(f"\n### Searching optimal parameters for {metric} ###")
        clf = GridSearchCV(
            ExtraTreesClassifier(random_state=42),
            parameter_grid,
            cv=5,
            scoring=metric
        )
        clf.fit(X_train, y_train)

        # Вивід результатів Grid Search
        print("\nGrid scores for the parameter grid:")
        for mean_score, params in zip(clf.cv_results_['mean_test_score'],
clf.cv_results_['params']):
            print(f"{params} --> {mean_score:.3f}")

        print("\nBest parameters:", clf.best_params_)

        # Вивід звіту на тестових даних
        y_pred = clf.predict(X_test)
        print(f"\nPerformance report for metric: {metric}\n")
        print(classification_report(y_test, y_pred))

        # Візуалізація класифікації на тестових даних
        visualize_classifier(
            clf.best_estimator_,
            X_test,
            y_test,
            f'Test Data Classification ({metric})',
            save_file=f'test_classification_{metric}.png'
        )

if __name__ == "__main__":
    main()
```

LR_5_task_4.py

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.datasets import fetch_california_housing
from sklearn.metrics import mean_squared_error, explained_variance_score
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

# --- Завантаження, перемішування та поділ даних ---
housing_data = fetch_california_housing()
X, y = shuffle(housing_data.data, housing_data.target, random_state=7)

# Розбиття на навчальний та тестовий набори (20% для тестування)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=7
)

# --- Побудова та навчання моделі AdaBoost ---
regressor = AdaBoostRegressor(
    estimator=DecisionTreeRegressor(max_depth=4),  # Базовий оцінювач - дерево глибиною 4
    n_estimators=400,                              # Кількість дерев
    random_state=7
)

regressor.fit(X_train, y_train)

# --- Оцінка ефективності регресора ---
y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)

print("\nADABOOST REGRESSOR")
print("Mean squared error =", round(mse, 2))
print("Explained variance score =", round(evs, 2))

# --- Вилучення та візуалізація важливості ознак ---
feature_importances = regressor.feature_importances_
feature_names = np.array(housing_data.feature_names)  # Перетворюємо у масив для індексації

# Нормалізація важливості ознак
feature_importances = 100.0 * (feature_importances / max(feature_importances))

# Сортування значень для графіка
index_sorted = np.flipud(np.argsort(feature_importances))
pos = np.arange(index_sorted.shape[0]) + 0.5

# Побудова стовпчастої діаграми
```

```
plt.figure(figsize=(12, 6))
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, feature_names[index_sorted], rotation=45, ha='right')
plt.ylabel('Relative Importance')
plt.title("Оцінка важливості ознак з використанням регресора AdaBoost")
plt.tight_layout()
plt.show()
```

LR_5_task_5.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import mean_absolute_error, accuracy_score
from sklearn.model_selection import train_test_split

# --- 1. Завантаження та попередня обробка даних ---

input_file = 'traffic_data.txt'
data = []

with open(input_file, 'r') as f:
    for line in f.readlines():
        items = line.strip().split(',')
        data.append(items)

data = np.array(data)

# --- 2. Кодування категоріальних ознак ---
label_encoders = []
X_encoded = np.empty(data.shape, dtype=int)

for i in range(data.shape[1]):
    # Перевіряємо, чи числовий стовпець
    try:
        X_encoded[:, i] = data[:, i].astype(int)
    except ValueError:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(data[:, i])
        label_encoders.append(le)

# Відокремлюємо ознаки та цільову змінну
X = X_encoded[:, :-1]  # ознаки
y = X_encoded[:, -1]   # кількість транспортних засобів

# --- 3. Розбиття на навчальний та тестовий набори ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)
```

```
# --- 4. Створення та навчання класифікатора Extra Trees ---
clf = ExtraTreesClassifier(
    n_estimators=200,    # кількість дерев
    max_depth=15,        # максимальна глибина дерев
    random_state=42
)
clf.fit(X_train, y_train)

# --- 5. Оцінка ефективності моделі ---
y_pred = clf.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae:.2f}")
print(f"Accuracy on test set: {accuracy:.2f}")

# --- 6. Прогноз для нової точки даних ---
test_datapoint = ['Saturday', '10:20', 'Atlanta', 'no']
test_encoded = []

le_index = 0
for i, item in enumerate(test_datapoint):
    try:
        test_encoded.append(int(item))
    except ValueError:
        test_encoded.append(int(label_encoders[le_index].transform([item])[0]))
        le_index += 1

test_encoded = np.array(test_encoded).reshape(1, -1)
predicted = clf.predict(test_encoded)[0]
print(f"Predicted traffic intensity: {predicted}")
```