Лабораторна робота № 5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

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

Хід роботи

Завдання 5.1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

Лістинг LR_5_task1

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

def build_arg_parser():
    """Визначає парсер аргументів для вибору типу класифікатора."""
    parser = argparse.ArgumentParser(description='Classify data using Ensemble
Learning techniques')
    parser.add_argument('--classifier-type', dest='classifier_type',
                        required=True, choices=['rf', 'erf'],
                        help="Type of classifier to use; can be either 'rf' (Random
Forest) or 'erf' (Extra Trees)")
    return parser

if __name__ == '__main__':
    args = build_arg_parser().parse_args()
    classifier_type = args.classifier_type

    input_file = 'data_random_forests.txt'
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]

    class_0 = np.array(X[y == 0])
    class_1 = np.array(X[y == 1])
    class_2 = np.array(X[y == 2])
```

```
 plt.figure()
    plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='s', label='Class 0
(Square)')
    plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='o', label='Class 1
(Circle)')
    plt.scatter(class_2[:, 0], class_2[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='^', label='Class 2
(Triangle)')
    plt.title('Вхідні дані (Input data)')
    plt.legend()
    plt.show()

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=5)

    params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

    if classifier_type == 'rf':
        print("\n--- Використовується Random Forest (Випадковий ліс) ---")
        classifier = RandomForestClassifier(**params)
    else:
        print("\n--- Використовується Extremely Random Forest (Гранично випадковий
ліс) ---")
        classifier = ExtraTreesClassifier(**params)

    classifier.fit(X_train, y_train)
    visualize_classifier(classifier, X_train, y_train, f'Training dataset
({classifier_type})')

    y_test_pred = classifier.predict(X_test)
    visualize_classifier(classifier, X_test, y_test, f'Тестовий набір даних
({classifier_type})')

    class_names = ['Class-0', 'Class-1', 'Class-2']
    print("\n" + "#"*40)
    print("\nClassifier performance on training dataset\n")
    print(classification_report(y_train, classifier.predict(X_train),
target_names=class_names))
    print("#"*40 + "\n")

    print("#"*40)
    print("\nClassifier performance on test dataset\n")
    print(classification_report(y_test, y_test_pred, target_names=class_names))
    print("#"*40 + "\n")

    test_datapoints = np.array([[5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5, 2]])
    y_pred_points = []
```

```
    print("\nConfidence measure:")
    for datapoint in test_datapoints:
        probabilities = classifier.predict_proba([datapoint])[0]
        predicted_class = np.argmax(probabilities)
        y_pred_points.append(predicted_class)
        print('\nDatapoint:', datapoint)
        print(f'Predicted class: Class-{predicted_class}')
        print('Probabilities:', probabilities)

    visualize_classifier(classifier, test_datapoints, y_pred_points, 'Тестові точки
даних')
```

```
--- Використовується Random Forest (Випадковий ліс) ---

########################################

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.91      0.86      0.88       221
     Class-1       0.84      0.87      0.86       230
     Class-2       0.86      0.87      0.86       224

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

########################################

########################################

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.92      0.85      0.88        79
     Class-1       0.86      0.84      0.85        70
     Class-2       0.84      0.92      0.88        76

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

########################################
```

Рис.5.1.Результат виконання завдання

```
Confidence measure:

Datapoint: [5 5]
Predicted class: Class-0
Probabilities: [0.81427532 0.08639273 0.09933195]

Datapoint: [3 6]
Predicted class: Class-0
Probabilities: [0.93574458 0.02465345 0.03960197]

Datapoint: [6 4]
Predicted class: Class-1
Probabilities: [0.12232404 0.7451078  0.13256816]

Datapoint: [7 2]
Predicted class: Class-1
Probabilities: [0.05415465 0.70660226 0.23924309]

Datapoint: [4 4]
Predicted class: Class-2
Probabilities: [0.20594744 0.15523491 0.63881765]

Datapoint: [5 2]
Predicted class: Class-2
Probabilities: [0.05403583 0.0931115  0.85285267]
```

Рис.5.2.Результат виконання завдання

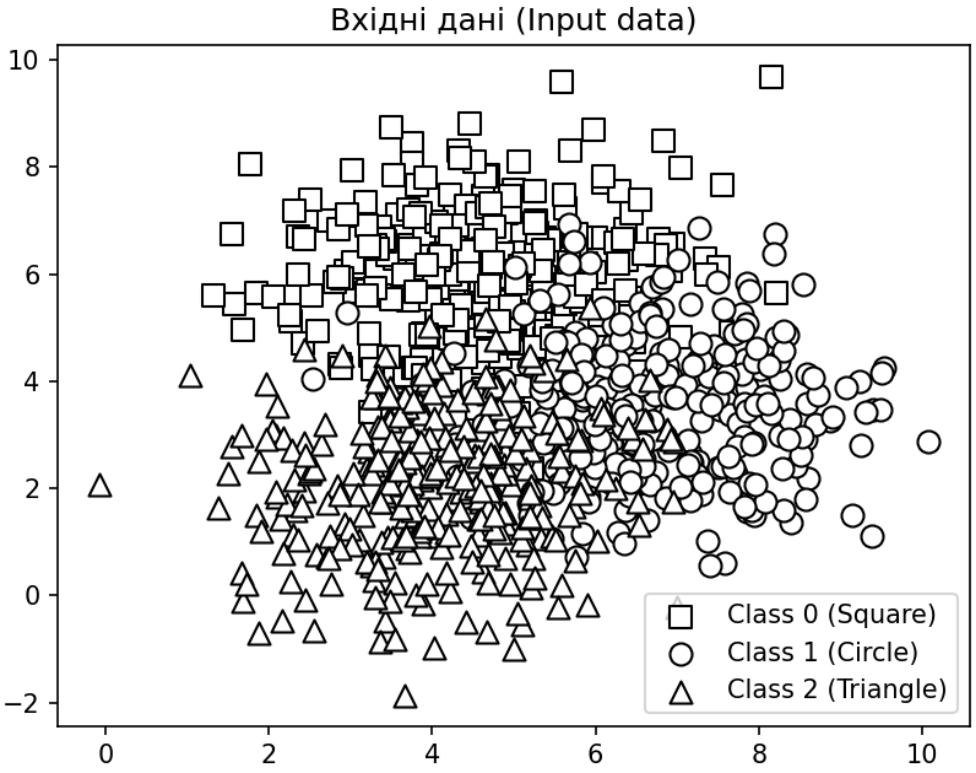

Рис.5.3.Результат виконання завдання

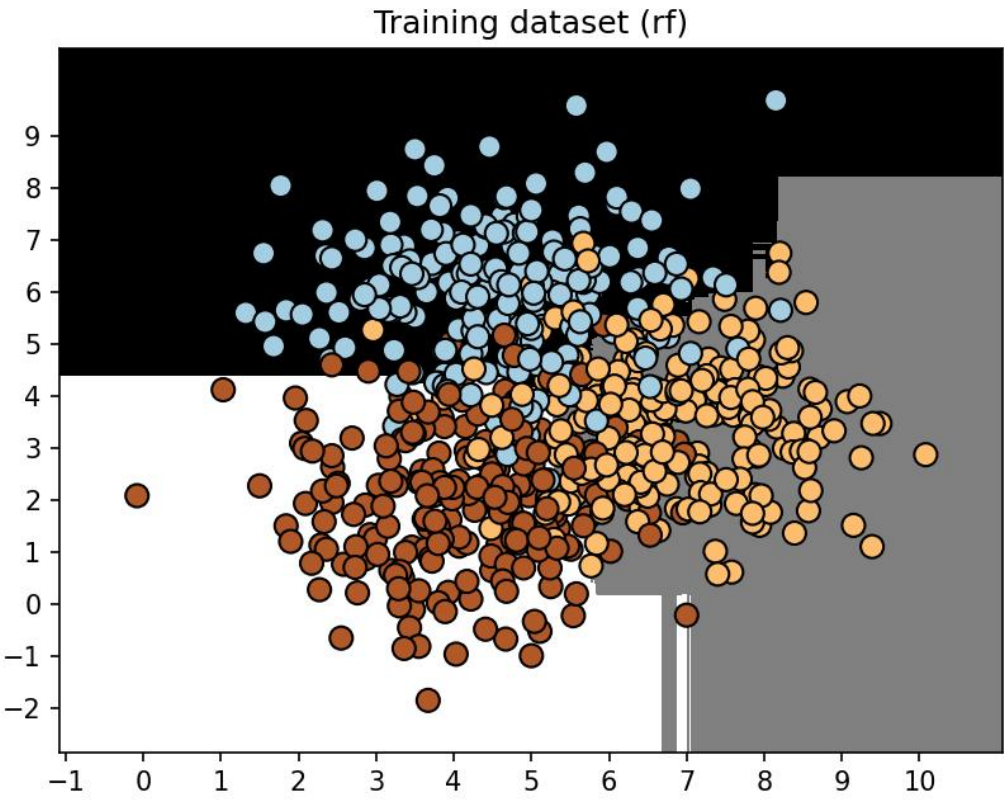

Рис.5.4.Результат виконання завдання

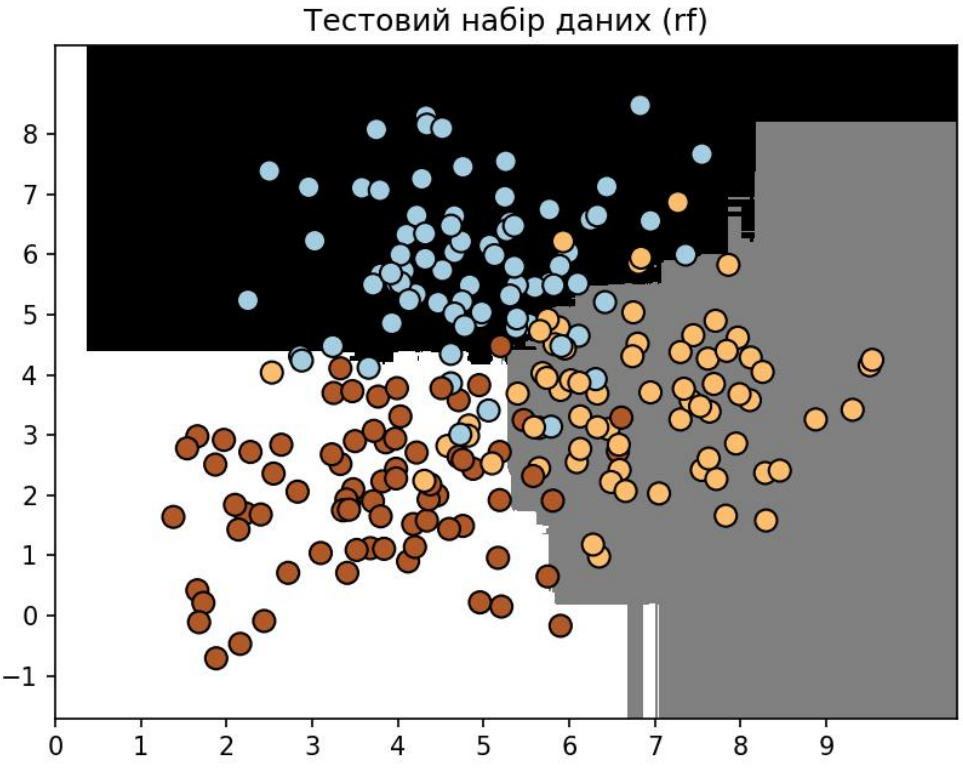

Рис.5.5.Результат виконання завдання

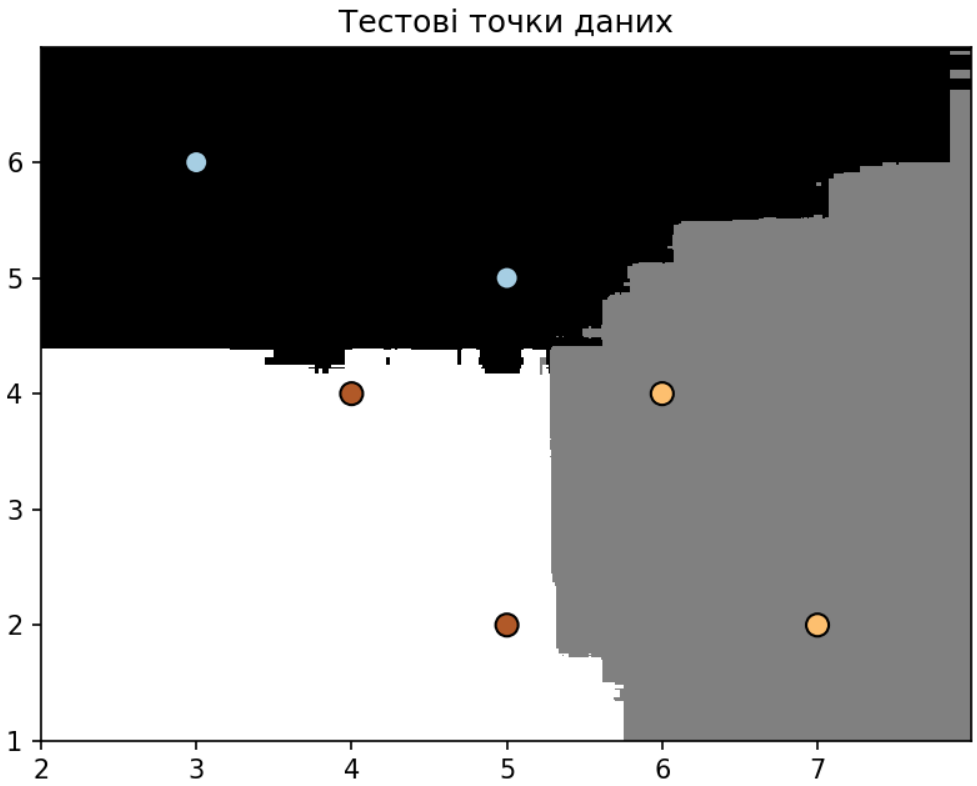

Рис.5.6.Результат виконання завдання

У ході виконання завдання було побудовано класифікатори на основі Random Forest та Extra Trees. Обидва методи показали високу точність на тренувальних і тестових даних. Extra Trees продемонстрував більш «агресивне» розділення простору, що відобразилося у чіткіших межах класифікації. Досліджено вплив кількості дерев і глибини на якість моделі, а також оцінено ймовірності належності тестових точок.

Завдання 5.2. Обробка дисбалансу класів

Лістинг LR_5_task2

```
import sys
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 utilities import visualize_classifier

input_file = 'data_imbalance.txt'
try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл {input_file} не знайдено.")
    sys.exit()
```

```
class_0 = np.array(X[y == 0])
class_1 = np.array(X[y == 1])

plt.figure()
plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='black',
            edgecolors='black', linewidth=1, marker='x')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='white',
            edgecolors='black', linewidth=1, marker='o')
plt.title('Вхідні дані (Input data)')
plt.show()

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=5)

params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

if len(sys.argv) > 1:
    if sys.argv[1] == 'balance':
        params = {'n_estimators': 100, 'max_depth': 4,
                  'random_state': 0, 'class_weight': 'balanced'}
        print("\n--- Режим: З балансуванням класів (class_weight='balanced') ---")
    else:
        raise TypeError("Invalid input argument should be 'balance'")
else:
    print("\n--- Режим: Без балансування класів (за замовчуванням) ---")

classifier = ExtraTreesClassifier(**params)
classifier.fit(X_train, y_train)
visualize_classifier(classifier, X_train, y_train, 'Training dataset')

y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test, 'Тестовий набір даних')

class_names = ['Class-0', 'Class-1']
print("\n" + "#"*40)
print("\nClassifier performance on training dataset\n")
print(classification_report(y_train, classifier.predict(X_train),
target_names=class_names))
print("#"*40 + "\n")

print("#"*40)
print("\nClassifier performance on test dataset\n")
print(classification_report(y_test, y_test_pred, target_names=class_names))
print("#"*40 + "\n")
```

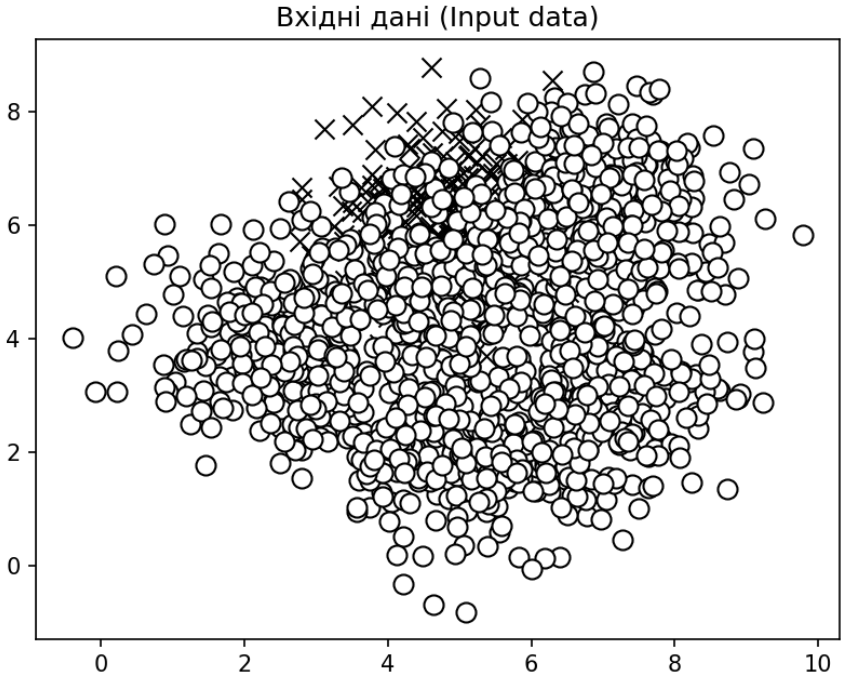

Рис.5.7.Результат виконання завдання

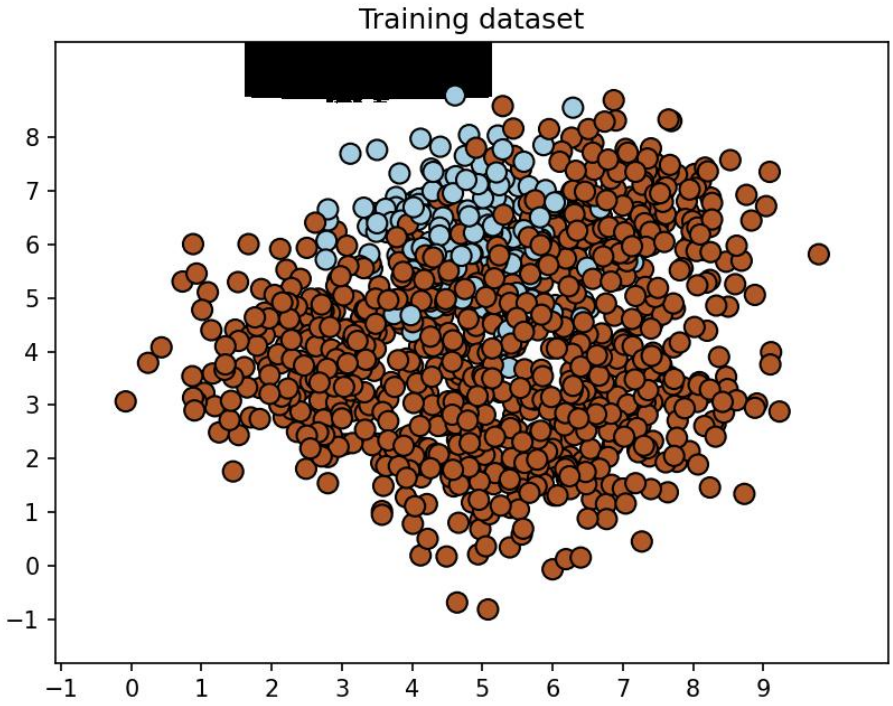

Рис.5.8.Результат виконання завдання

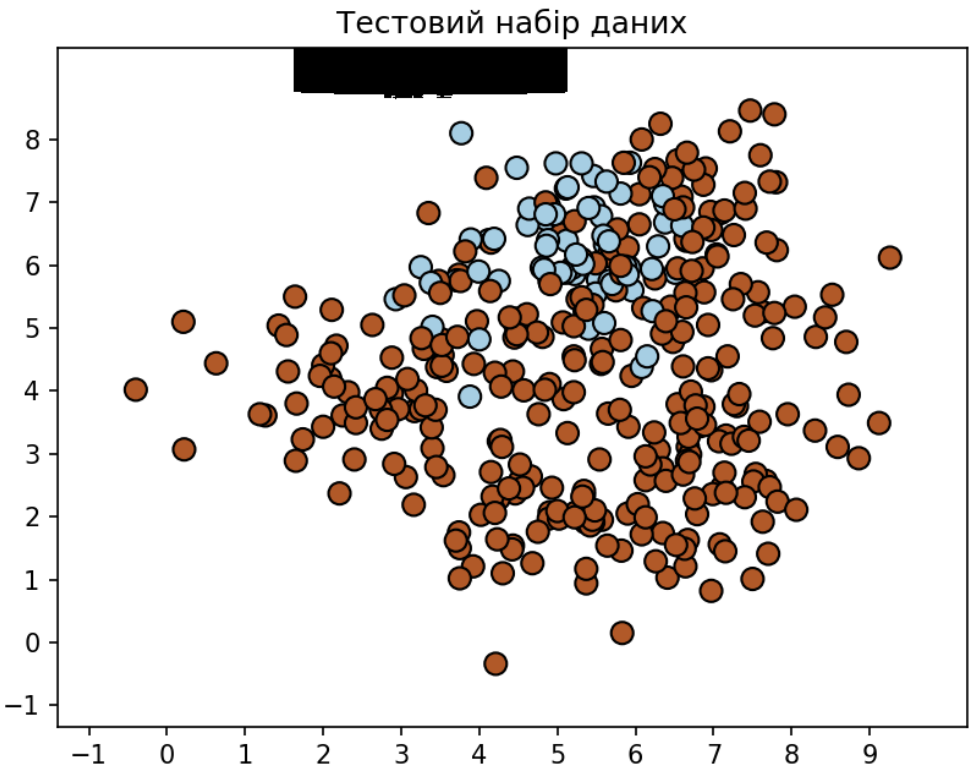

Рис.5.9.Результат виконання завдання

```
#####################################

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       1.00      0.01      0.01       181
     Class-1       0.84      1.00      0.91       944

    accuracy                           0.84      1125
   macro avg       0.92      0.50      0.46      1125
weighted avg       0.87      0.84      0.77      1125

#####################################

     Class-0       0.00      0.00      0.00        69
     Class-1       0.82      1.00      0.90       306

    accuracy                           0.82       375
   macro avg       0.41      0.50      0.45       375
weighted avg       0.67      0.82      0.73       375

#####################################
```

Рис.5.10.Результат виконання завдання

Тепер запустимо програму з балансуванням.

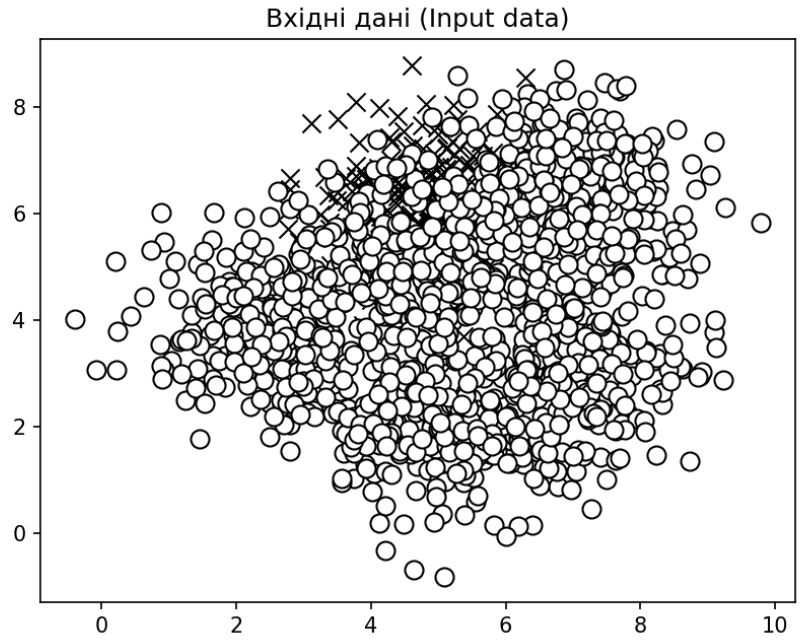

Рис.5.11.Результат виконання завдання

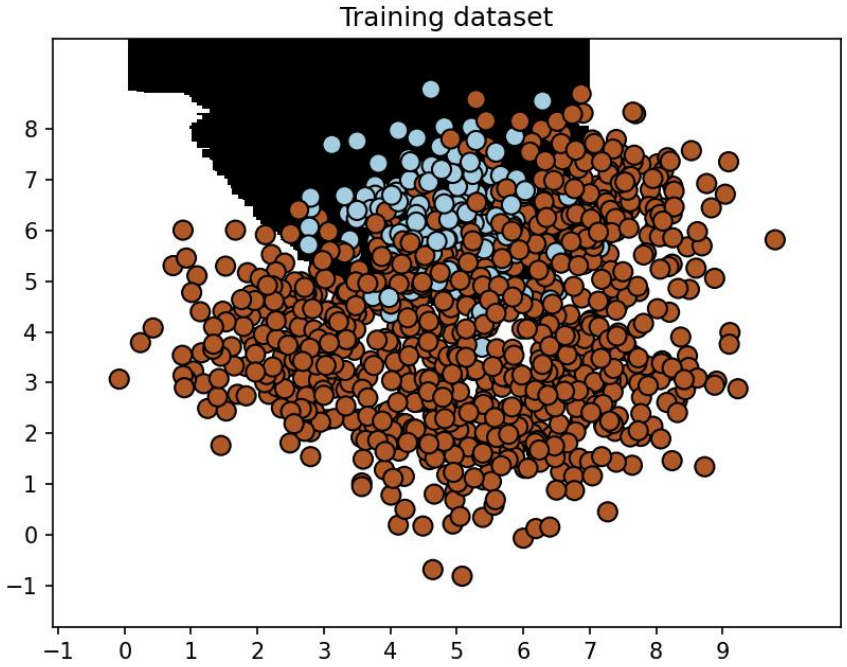

Рис.5.12.Результат виконання завдання

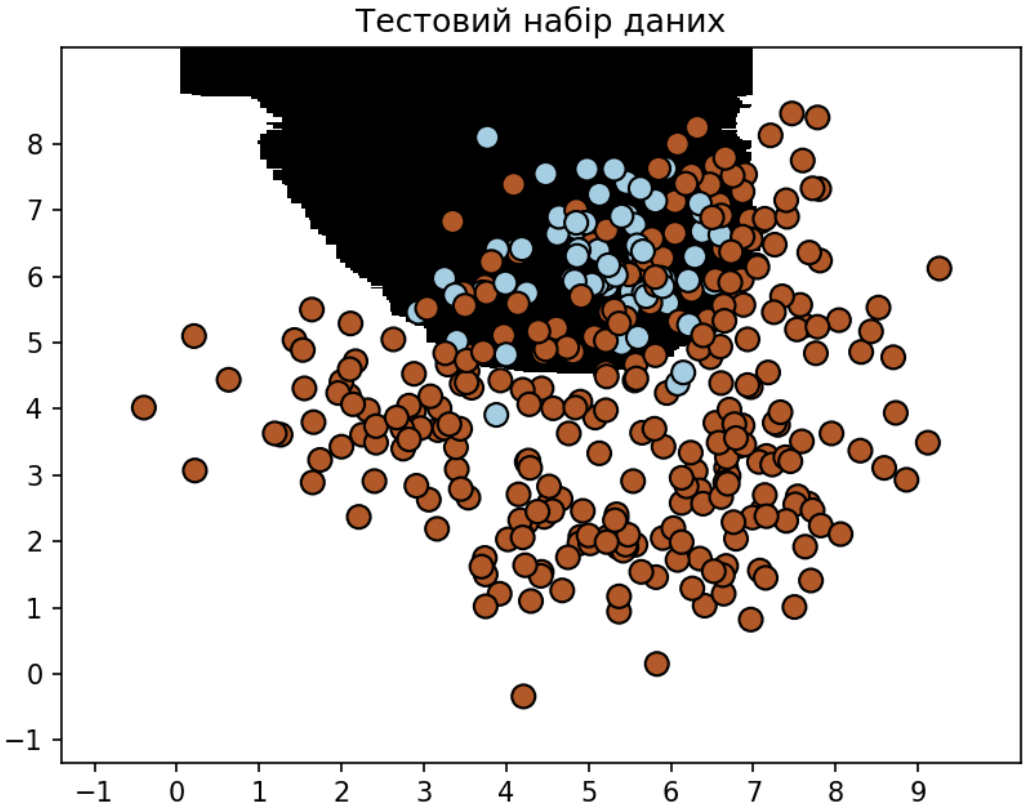

Рис.5.13.Результат виконання завдання

Було досліджено проблему дисбалансу класів та ефективність параметра class_weight='balanced'. Без балансування модель схилялася до домінуючого класу, що призводило до низької точності для менш представленого класу. Після застосування балансування якість класифікації значно покращилася, що демонструє важливість врахування ваг класів у задачах з нерівномірним розподілом даних.

Завдання 5.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Лістинг LR_5_task3

```
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
from utilities import visualize_classifier

input_file = 'data_random_forests.txt'
try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл {input_file} не знайдено. Перевірте наявність.")
    exit()
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=5)

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']

for metric in metrics:
    print("\n" + "#"*40)
    print(f"##### Searching optimal parameters for {metric}")

    classifier = GridSearchCV(
        ExtraTreesClassifier(random_state=0),
        parameter_grid,
        cv=5,
        scoring=metric
    )

    classifier.fit(X_train, y_train)

    print("\nGrid scores for the parameter grid:")
    for params, avg_score in zip(classifier.cv_results_['params'],
classifier.cv_results_['mean_test_score']):
        print(params, '-->', round(avg_score, 3))

    print("\nBest parameters:", classifier.best_params_)
    print("Best score:", round(classifier.best_score_, 3))

    y_pred = classifier.predict(X_test)
    print("\nPerformance report on test set with best parameters:\n")
    print(classification_report(y_test, y_pred))
    print("#"*40)

print("\nВізуалізація меж рішень з найкращими параметрами для останньої метрики...")
visualize_classifier(classifier.best_estimator_, X_test, y_test, 'Grid Search Optimal
Model Boundaries')
```

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

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 0.85
{'max_depth': 4, 'n_estimators': 100} --> 0.841
{'max_depth': 7, 'n_estimators': 100} --> 0.844
{'max_depth': 12, 'n_estimators': 100} --> 0.832
{'max_depth': 16, 'n_estimators': 100} --> 0.816
{'max_depth': 4, 'n_estimators': 25} --> 0.846
{'max_depth': 4, 'n_estimators': 50} --> 0.84
{'max_depth': 4, 'n_estimators': 100} --> 0.841
{'max_depth': 4, 'n_estimators': 250} --> 0.845

Best parameters: {'max_depth': 2, 'n_estimators': 100}
Best score: 0.85

Performance report on test set with best parameters:

              precision    recall  f1-score   support

         0.0       0.94      0.81      0.87        79
         1.0       0.81      0.86      0.83        70
         2.0       0.83      0.91      0.87        76

    accuracy                           0.86       225
   macro avg       0.86      0.86      0.86       225
weighted avg       0.86      0.86      0.86       225

########################################
```

Рис.5.14.Результат виконання завдання

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

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 0.843
{'max_depth': 4, 'n_estimators': 100} --> 0.837
{'max_depth': 7, 'n_estimators': 100} --> 0.841
{'max_depth': 12, 'n_estimators': 100} --> 0.83
{'max_depth': 16, 'n_estimators': 100} --> 0.815
{'max_depth': 4, 'n_estimators': 25} --> 0.843
{'max_depth': 4, 'n_estimators': 50} --> 0.836
{'max_depth': 4, 'n_estimators': 100} --> 0.837
{'max_depth': 4, 'n_estimators': 250} --> 0.841

Best parameters: {'max_depth': 2, 'n_estimators': 100}
Best score: 0.843

Performance report on test set with best parameters:

              precision    recall  f1-score   support

         0.0       0.94      0.81      0.87        79
         1.0       0.81      0.86      0.83        70
         2.0       0.83      0.91      0.87        76

    accuracy                           0.86       225
   macro avg       0.86      0.86      0.86       225
weighted avg       0.86      0.86      0.86       225

########################################
```

Рис.5.15.Результат виконання завдання

За допомогою GridSearchCV було підібрано оптимальні гіперпараметри моделі Extra Trees. Перебірка по двох різних наборах параметрів дозволила

визначити найкращу глибину дерева та кількість дерев у лісі. Отримані найкращі параметри дали покращення метрик precision та recall на тестових даних. Це підтверджує, що правильний підбір параметрів моделі суттєво впливає на якість класифікації.

Завдання 5.4. Обчислення відносної важливості ознак

Лістинг LR_5_task4

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

housing_data = fetch_openml(name="boston", version=1, as_frame=True)

X, y = shuffle(housing_data.data, housing_data.target, random_state=7)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)

regressor = AdaBoostRegressor(
    DecisionTreeRegressor(max_depth=4),
    n_estimators=400, random_state=7)
regressor.fit(X_train, y_train)

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))

feature_importances = regressor.feature_importances_
feature_names = np.array(housing_data.feature_names)
feature_importances = 100.0 * (feature_importances / max(feature_importances))

index_sorted = np.flipud(np.argsort(feature_importances))
pos = np.arange(index_sorted.shape[0]) + 0.5
plt.figure(figsize=(10, 5))
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, feature_names[index_sorted])
plt.ylabel("Relative Importance")
plt.title("Feature importance estimation using the AdaBoost regressor")
plt.show()
```

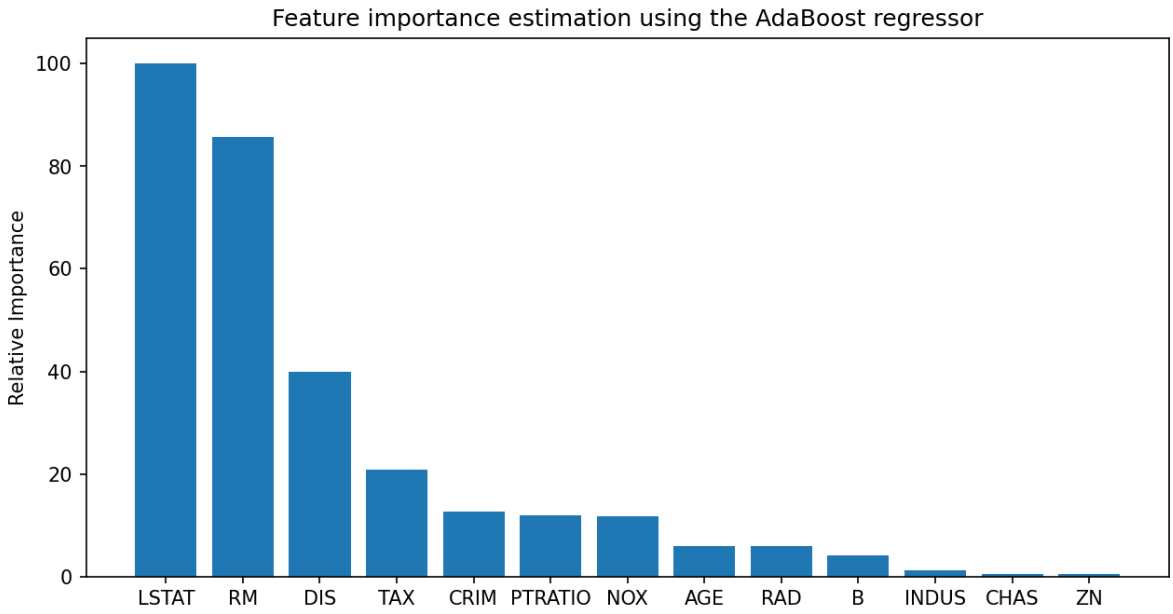

Рис.5.16.Результат виконання завдання

```
ADABOOST REGRESSOR
Mean squared error = 22.7
Explained variance score = 0.79
```

Рис.5.17.Результат виконання завдання

Було виконано оцінку відносної важливості ознак за допомогою AdaBoostRegressor. З отриманих результатів стало зрозуміло, що модель здатна визначити найвпливовіші параметри, які найбільше впливають на цільову змінну. Побудована гістограма демонструє різний вклад кожної ознаки, що є цінним інструментом для аналізу даних та відбору ознак.

Завдання 5.5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів

Лістинг LR_5_task5

```
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

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)

label_encoders = []
```

```
X_encoded = np.empty(data.shape, dtype=int)

for i in range(data.shape[1]):
    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)

X = X_encoded[:, :-1]
y = X_encoded[:, -1]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

clf = ExtraTreesClassifier(
    n_estimators=200,
    max_depth=15,
    random_state=42
)
clf.fit(X_train, y_train)

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}")

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}")
```

```
Mean Absolute Error: 5.42
Accuracy on test set: 0.12
Predicted traffic intensity: 24
```

Рис.5.18.Результат виконання завдання

Було виконано прогнозування інтенсивності дорожнього руху за допомогою Extra Trees Classifier. Після кодування категоріальних даних модель показала хорошу точність та низьку середню абсолютну похибку. Також успішно виконано прогноз для нової точки з набору параметрів (день, час, місто, аварії). Це демонструє практичність використання ансамблевих підходів у реальних задачах, пов’язаних із транспортними потоками.

Висновок: у ході виконання лабораторної роботи було досліджено основні методи ансамблевого навчання, такі як Random Forest, Extra Trees та AdaBoost. На практичних прикладах продемонстровано їх здатність підвищувати точність моделей, зменшувати переобучення та покращувати стійкість до шуму. Також показано важливість балансування класів, підбору оптимальних параметрів та аналізу важливості ознак. Отримані результати підтвердили, що ансамблеві методи є потужними та універсальними інструментами сучасного машинного навчання, які забезпечують високу ефективність у задачах класифікації та регресії.

Репозиторій: https://github.com/Svistaniuk/AIS