Лабораторна робота №5

ДОСЛІДЖЕННЯ 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 коду:

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

# Argument parser
def build_arg_parser():
    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' or 'erf'")
    return parser

if __name__=='__main__':
    # Parse the input arguments
    args = build_arg_parser().parse_args()
    classifier_type = args.classifier_type

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

    # Separate input data into three classes based on labels
    class_0 = np.array(X[y==0])
    class_1 = np.array(X[y==1])
    class_2 = np.array(X[y==2])

    # Visualize input data
    plt.figure()
    plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='s')
    plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='o')
```

```
plt.scatter(class_2[:, 0], class_2[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='^')
plt.title('Input data')

# Split data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=5)

# Ensemble Learning classifier
params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}
if classifier_type == 'rf':
    classifier = RandomForestClassifier(**params)
else:
    classifier = ExtraTreesClassifier(**params)

classifier.fit(X_train, y_train)
visualize_classifier(classifier, X_train, y_train)

y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test)

# Evaluate classifier performance
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")

# Compute confidence
test_datapoints = np.array([[5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5, 2]])

print("\nConfidence measure:")
for datapoint in test_datapoints:
    probabilities = classifier.predict_proba([datapoint])[0]
    predicted_class = 'Class-' + str(np.argmax(probabilities))
    print('\nDatapoint:', datapoint)
    print('Predicted class:', predicted_class)

# Visualize the datapoints
visualize_classifier(classifier, test_datapoints, [0]*len(test_datapoints))

plt.show()
```

Результат виконання:

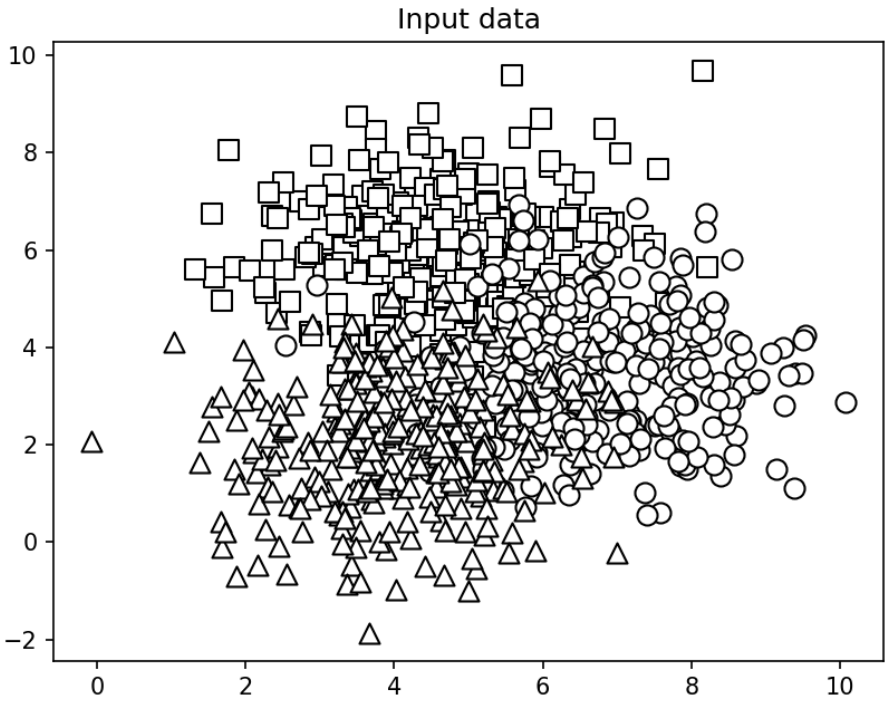

Рисунок 1 - Результат виконання програми (візуалізація введених даних)

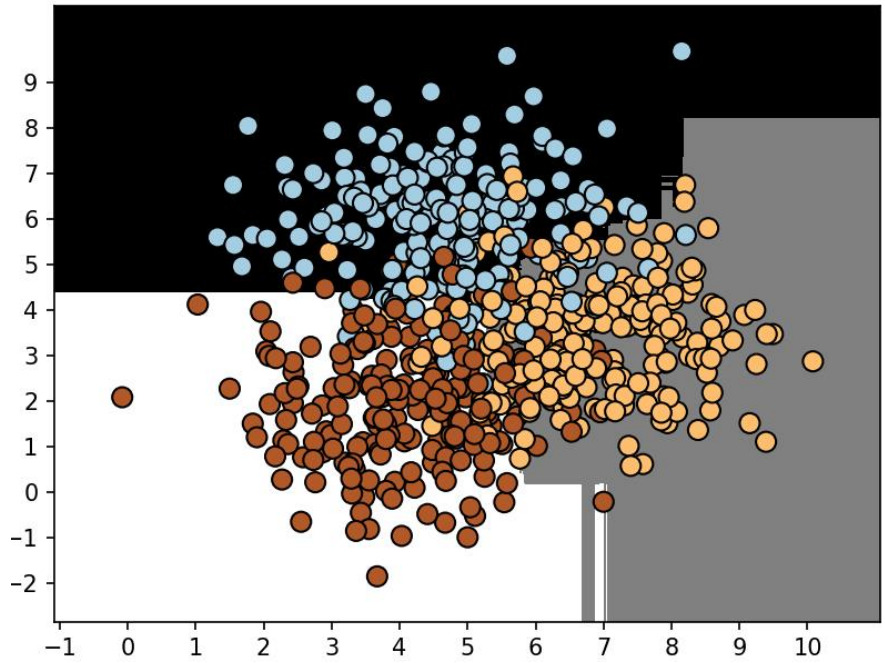

Рисунок 2 - Результат виконання програми (класифікатор на основі випадкового лісу для навчальних даних)

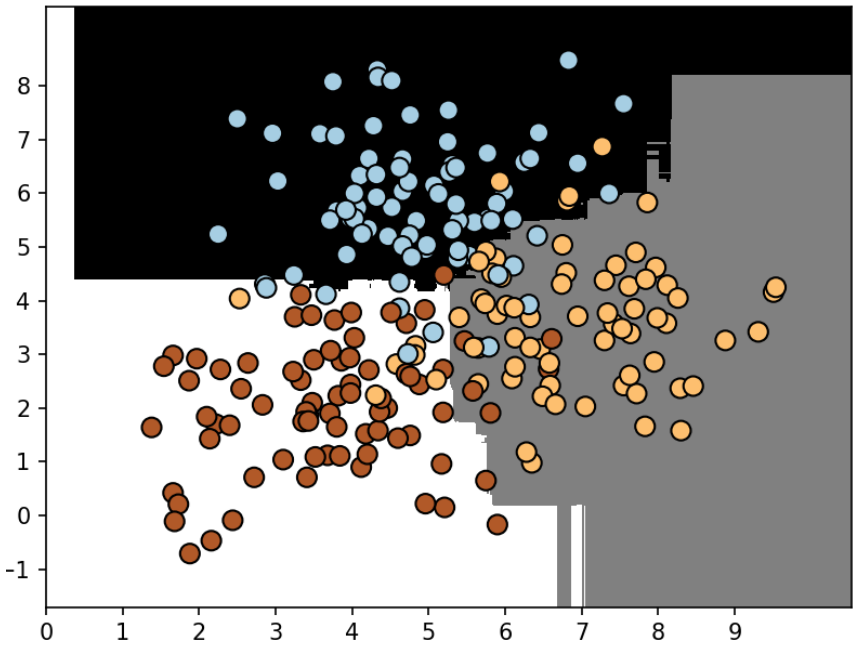

Рисунок 3 - Результат виконання програми (класифікатор на основі випадкового лісу для тестових даних)

```
Terminal:  Local
(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5> python random_forests.py --classifier-type rf

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

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

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

(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5>
```

Рисунок 4 - Результат виконання програми

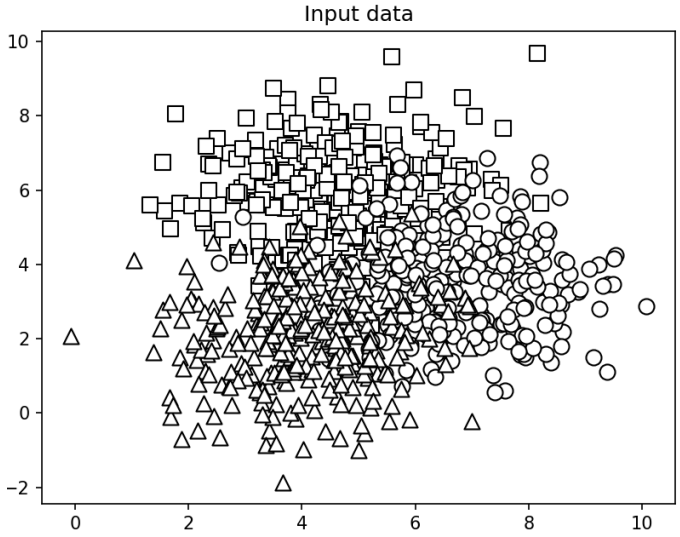

Рисунок 5 - Результат виконання програми (візуалізація введених даних)

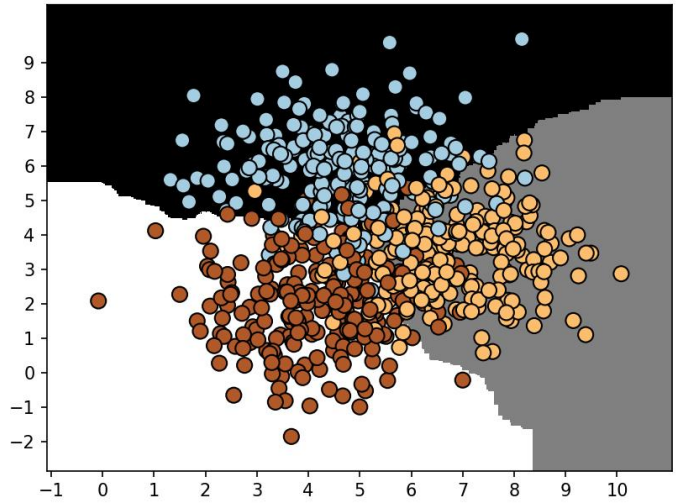

Рисунок 6 - Результат виконання програми (класифікатор на основі граничного випадкового лісу для навчальних даних)

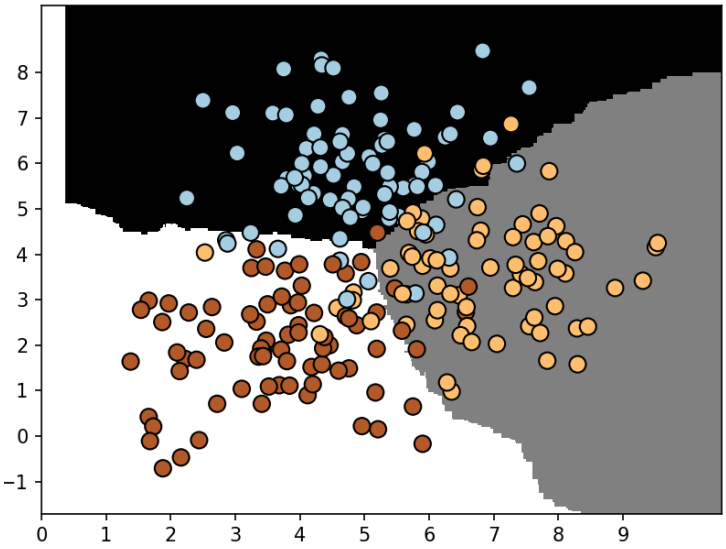

Рисунок 7 - Результат виконання програми (класифікатор на основі граничного випадкового лісу для тестових даних)

```
(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5> python random_forests.py --classifier-type erf

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

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.89      0.83      0.86       221
     Class-1       0.82      0.84      0.83       230
     Class-2       0.83      0.86      0.85       224

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

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

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

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.92      0.85      0.88        79
     Class-1       0.84      0.84      0.84        70
     Class-2       0.85      0.92      0.89        76

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

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

(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5>
```

Рисунок 8 - Результат виконання програми

Оцінка мір достовірності прогнозів

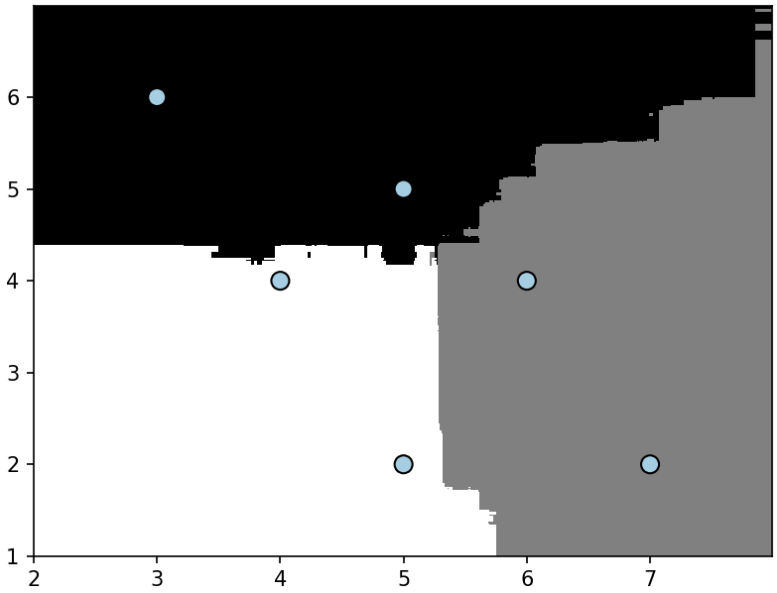

Рисунок 9 - Результат виконання програми на основі випадкового лісу

```
Terminal:  Local
(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5> python random_forests.py --classifier-type rf

Confidence measure:

Datapoint: [5 5]
Predicted class: Class-0

Datapoint: [3 6]
Predicted class: Class-0

Datapoint: [6 4]
Predicted class: Class-1

Datapoint: [7 2]
Predicted class: Class-1

Datapoint: [4 4]
Predicted class: Class-2

Datapoint: [5 2]
Predicted class: Class-2
(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5>
```

Рисунок 10 - Результат виконання програми на основі випадкового лісу

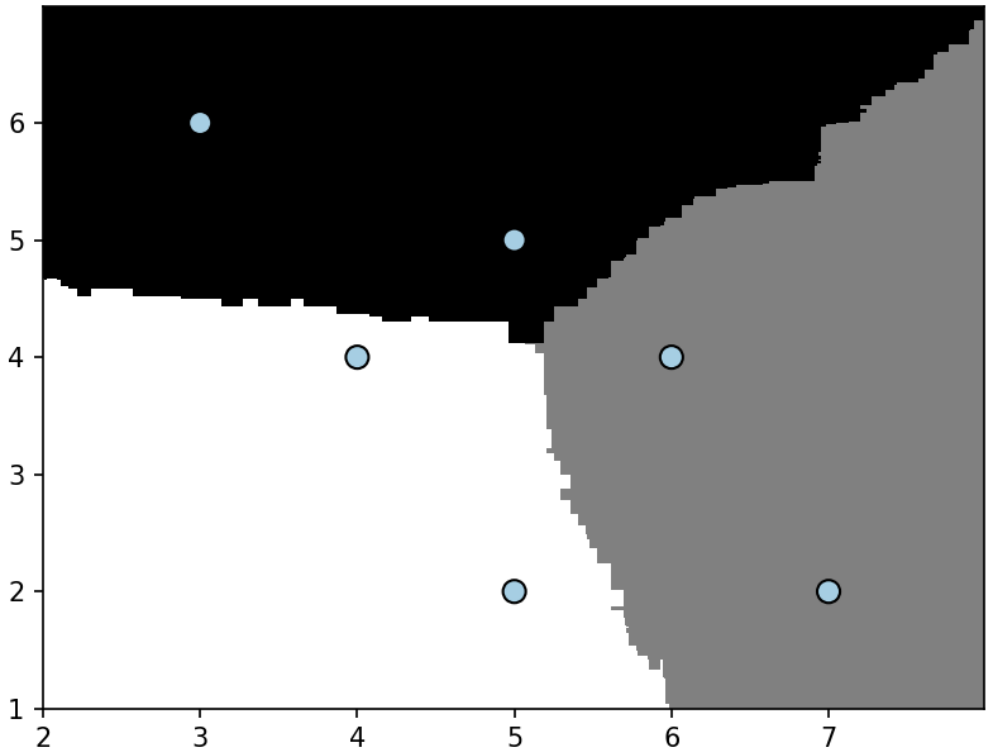

Рисунок 11 - Результат виконання програми на основі граничного випадкового лісу

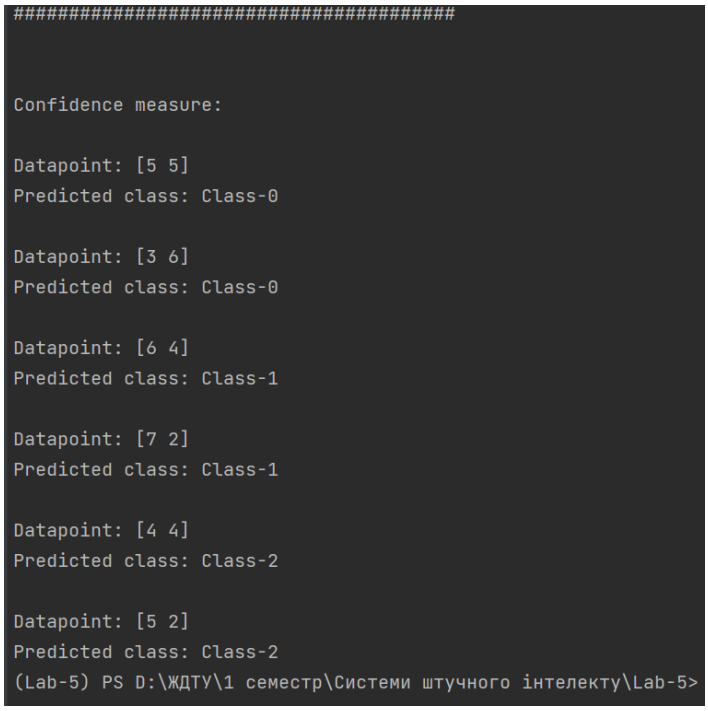

Рисунок 12 - Результат виконання програми на основі граничного випадкового лісу

1. На тренувальному наборі даних

Extra Random Forest:

- **Accuracy**: 85%
- **Precision (середнє)**: 85%
- **Recall (середнє)**: 85%
- **F1-score (середнє)**: 85%

Random Forest:

- **Accuracy**: 87%
- **Precision (середнє)**: 87%
- **Recall (середнє)**: 87%
- **F1-score (середнє)**: 87%

Висновок: Random Forest показує трохи кращі результати на тренувальному наборі, зокрема, на 2% вищу точність.

2. На тестовому наборі даних

Extra Random Forest:

- **Accuracy**: 87%
- **Precision (середнє)**: 87%

- **Recall (середнє)**: 87%
- **F1-score (середнє)**: 87%

Random Forest:

- **Accuracy**: 87%
- **Precision (середнє)**: 87%
- **Recall (середнє)**: 87%
- **F1-score (середнє)**: 87%

Висновок: На тестовому наборі обидва класифікатори показують однакову точність (87%) та інші метрики. Це вказує на те, що обидва моделі мають схожу ефективність у прогнозуванні на нових даних.

3. Прогнозування на нових точках (Confidence measure)

Обидва класифікатори дають однакові прогнози для тестових точок:

- [5, 5] -> Class-0
- [3, 6] -> Class-0
- [6, 4] -> Class-1
- [7, 2] -> Class-1
- [4, 4] -> Class-2
- [5, 2] -> Class-2

Загальний висновок

1. Random Forest показує трохи кращі результати на тренувальному наборі, але на тестовому наборі метрики однакові для обох класифікаторів.
2. Вибір між Random Forest і Extra Random Forest залежить від задачі:
    - Якщо потрібна краща узгодженість на тренувальних даних - Random Forest.
    - Якщо треба уникнути перенавчання через додаткову рандомізацію в розщепленні вузлів, Extra Random Forest може бути більш стійким до перенавчання.

Обидва підходи мають свої переваги, і можна використовувати обидва для порівняння залежно від реальних даних.

Завдання 5.2 Обробка дисбалансу класів

Лістинг коду:

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

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

# Поділ вхідних даних на два класи на підставі міток
class_0 = np.array(X[y == 0])
class_1 = np.array(X[y == 1])

# Візуалізація вхідних даних
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('Input data')

# Розбиття даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=5)

# Класифікатор на основі гранично випадкових лісів
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'}
    else:
        raise TypeError("Invalid input argument; should be 'balance'")

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

# Передбачимо та візуалізуємо результат для тестового набору даних
y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test)

# Обчислення показників ефективності класифікатора
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")
plt.show()
```

Результат виконання програми:

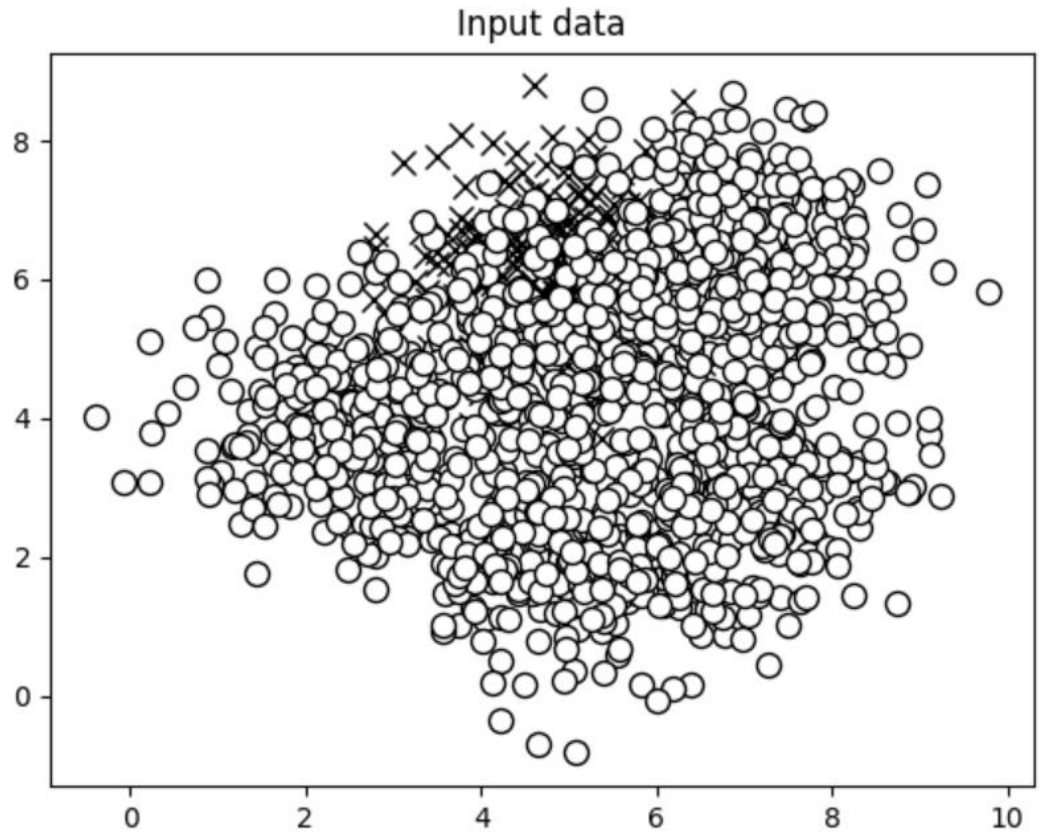

Рисунок 13 – Графік вхідних даних

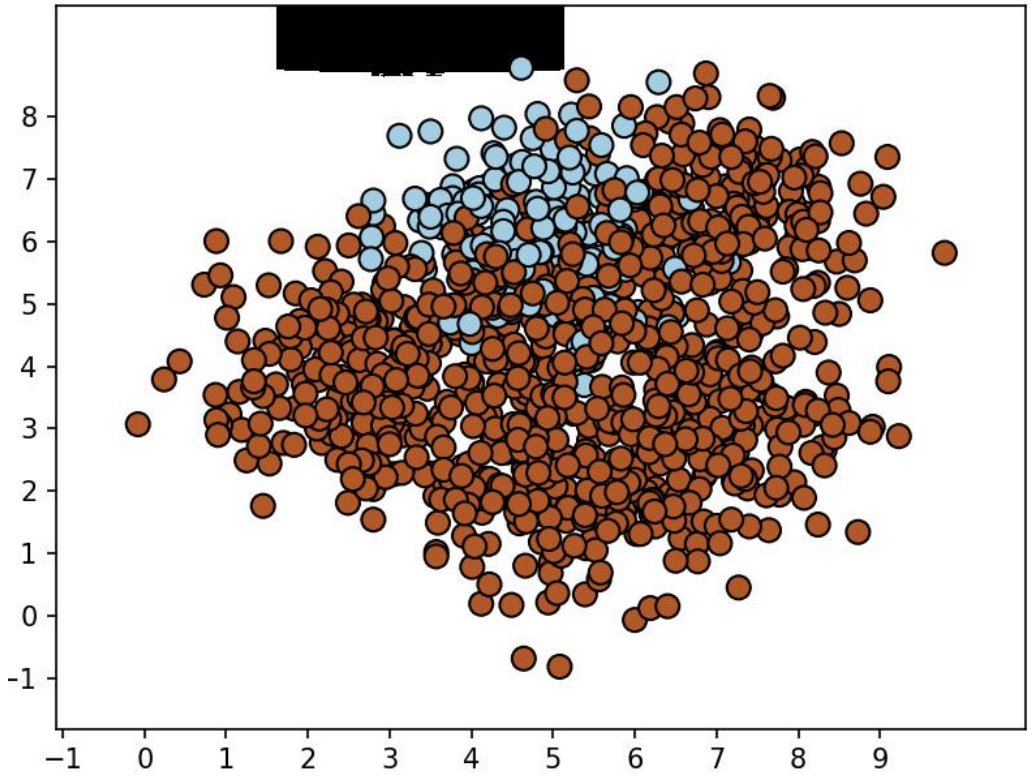

Рисунок 14 – Графік даних класифікатора для навчального набору

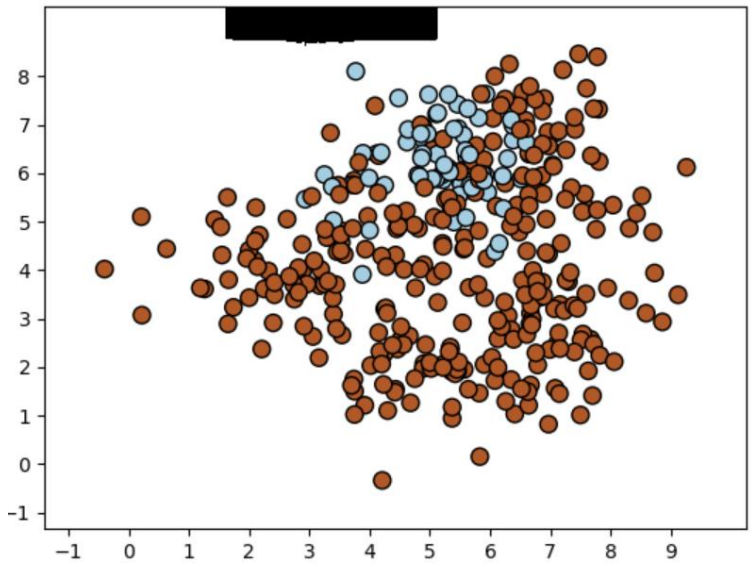

Рисунок 15 – Графік даних класифікатора для тестового набору

```
Terminal:  Local
(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5> python -W ignore .\LR_5_task_2.py

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

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

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

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

Classifier performance on test dataset

              precision    recall  f1-score   support

     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

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

(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5>
```

Рисунок 16 – Результат виконання програми

На основі результатів класифікації з використанням моделі випадкових лісів за умов дисбалансу класів, можна зробити такі висновки:

1. Результати на тренувальній вибірці

- **Class-0 (менш представлений клас):**
  - **Precision:** 1.00, але **Recall:** лише 0.01. Це вказує на те, що модель майже не знаходить прикладів цього класу.
  - **F1-score:** 0.01 — дуже низький показник.
- **Class-1 (переважний клас):**
  - **Precision:** 0.84, а **Recall:** 1.00. Модель добре знаходить приклади класу, але може включати хибнопозитивні значення.
  - **F1-score:** 0.91 — добрий результат.
- Загальна **accuracy:** 0.84, але:
  - **Macro avg F1-score:** 0.46 — значно нижчий, що вказує на суттєвий дисбаланс у продуктивності для різних класів.
  - **Weighted avg F1-score:** 0.77 — відображає перекіс у бік Class-1.

2. Результати на тестовій вибірці

- **Class-0:**
  - Precision, Recall, F1-score — 0.00. Модель не змогла розпізнати цей клас на тестовій вибірці.
- **Class-1:**
  - **Precision:** 0.82, **Recall:** 1.00, **F1-score:** 0.90. Модель майже ідеально працює з переважним класом.
- Загальна **accuracy:** 0.82, але:
  - **Macro avg F1-score:** 0.45 — дуже низький, через провал у класифікації Class-0.
  - **Weighted avg F1-score:** 0.73 — прийнятний рівень, але знову ж таки через перекіс

Завдання 5.3 Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Лістинг коду:

```
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 sklearn.metrics import classification_report
from utilities import visualize_classifier

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

# Розбиття даних на три класи на підставі міток
class_0 = np.array(X[y == 0])
class_1 = np.array(X[y == 1])
class_2 = np.array(X[y == 2])

# Розбиття даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.25,
random_state=5)

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

#Визначаємо метричні характеристики
metrics = ['precision_weighted', 'recall_weighted']

#Сітковий пошук
for metric in metrics:
    print("\n##### Searching optimal parameters for", metric)

    classifier = GridSearchCV(
        ExtraTreesClassifier(random_state=0),
        parameter_grid, cv=5, scoring=metric)
    classifier.fit(X_train, y_train)

    # Вивід оцінки для кожної комбінації параметрів
    print("\nGrid scores for the parameter grid:")
    for i in range(len(classifier.cv_results_['params'])):
        print(
            classifier.cv_results_['params'][i],
            '-->',
            round(classifier.cv_results_['mean_test_score'][i], 3)
        )

    #Результати роботи класифікатора
    y_pred = classifier.predict(X_test)
    print("\nPerformance report:\n")
    print(classification_report(y_test, y_pred))
```

Результат виконання програми:

```
Terminal:  Local
##### Searching optimal parameters for recall_weighted
(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5> python -W ignore .\LR_5_task_3.py

##### 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

Performance report:

              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


##### 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

Performance report:

              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

(Lab-5) PS D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-5>
```

Рисунок 17 – Результат виконання програми

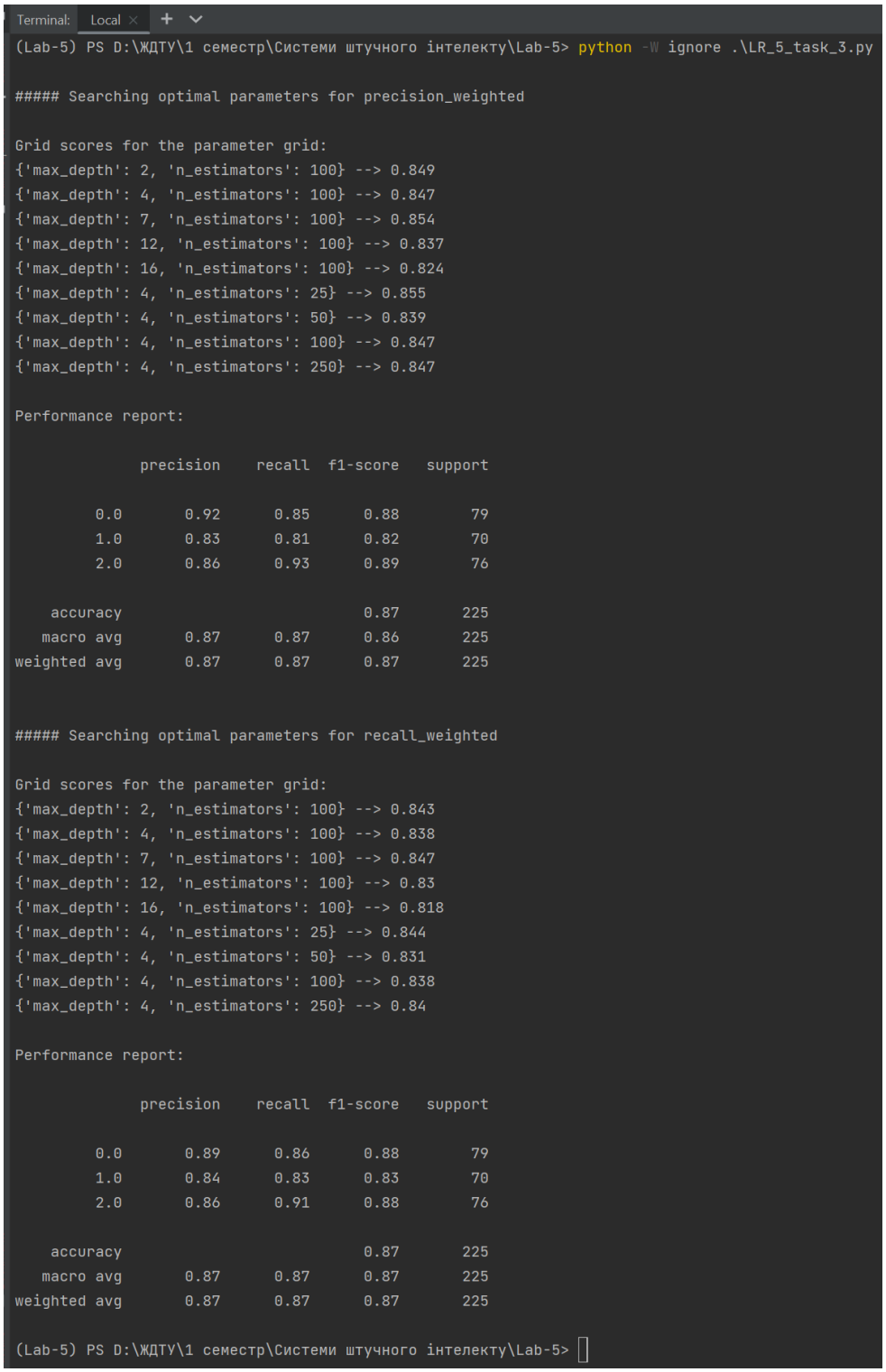

Рисунок 18 – Результат виконання програми (після зміни параметрів)

1. **Результати для метрики precision_weighted:**

- Найкраща комбінація параметрів: max_depth = 2 і n_estimators = 100, з оцінкою 0.85.
- В цілому, параметри з max_depth = 2 та різними значеннями n_estimators (особливо 25, 50 і 100) показали найкращі результати для цієї метрики.

- Результати класифікації показують високі показники точності та відсоткові оцінки для кожного класу, з середньою точністю (precision) 0.86.

2. **Результати для метрики recall_weighted:**

- Найкраща комбінація параметрів: max_depth = 2 і n_estimators = 100, з оцінкою 0.843.
- Результати для recall показують схожі тенденції, де комбінація з max_depth = 2 і різними значеннями n_estimators також дає кращі результати, хоча з трохи нижчими значеннями, порівняно з precision.
- Загальний результат для recall схожий: висока точність та схоже розподілення по класах, з середнім recall 0.86.

3. Основні спостереження:

- Обидві метрики (precision і recall) мають схожі результати, з найкращими комбінаціями параметрів, що включають max_depth = 2 і n_estimators = 100.
- Незважаючи на відмінності в метриках, модель показала хороші загальні результати для всіх класів з високою точністю і recall.

Після зміни параметрів **random_state** та **cv** деякі показники для сітки покращились, а деякі погіршились.

Завдання 5.4 Обчислення відносної важливості ознак

Лістинг коду:

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

# datasets.load_boston() Вилучино в нових версіях
# Завантаження даних із цінами на нерухомість
housing_data = datasets.fetch_california_housing()

# Перемішування даних
X, y = shuffle(housing_data.data, housing_data.target, random_state=7)

# Розбиття даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)
```

```
# Модель на основі регресора AdaBoost
regressor = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4),
                              n_estimators=400, random_state=7)
regressor.fit(X_train, y_train)

# Обчислення показників ефективності регресора AdaBoost
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 важливості ознак
feature_importances = regressor.feature_importances_
feature_names = np.array(housing_data.feature_names)

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

# Сортування та перестановка значень
index_sorted = np.flipud(np.argsort(feature_importances))

# Розміщення міток уздовж осі X
pos = np.arange(index_sorted.shape[0]) + 0.5

# Побудова стовпчастої діаграми (модернізована, бо по осі х погано
відображалося)
plt.figure(figsize=(10, 6))  # Збільшення розміру фігури
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, feature_names[index_sorted], rotation=45, ha='right')  #
Обертання міток
plt.ylabel('Relative Importance')
plt.title('Feature importance using AdaBoost regressor')
plt.tight_layout()  # Оптимізація розташування компонентів
plt.show()
```

Результат виконання програми:

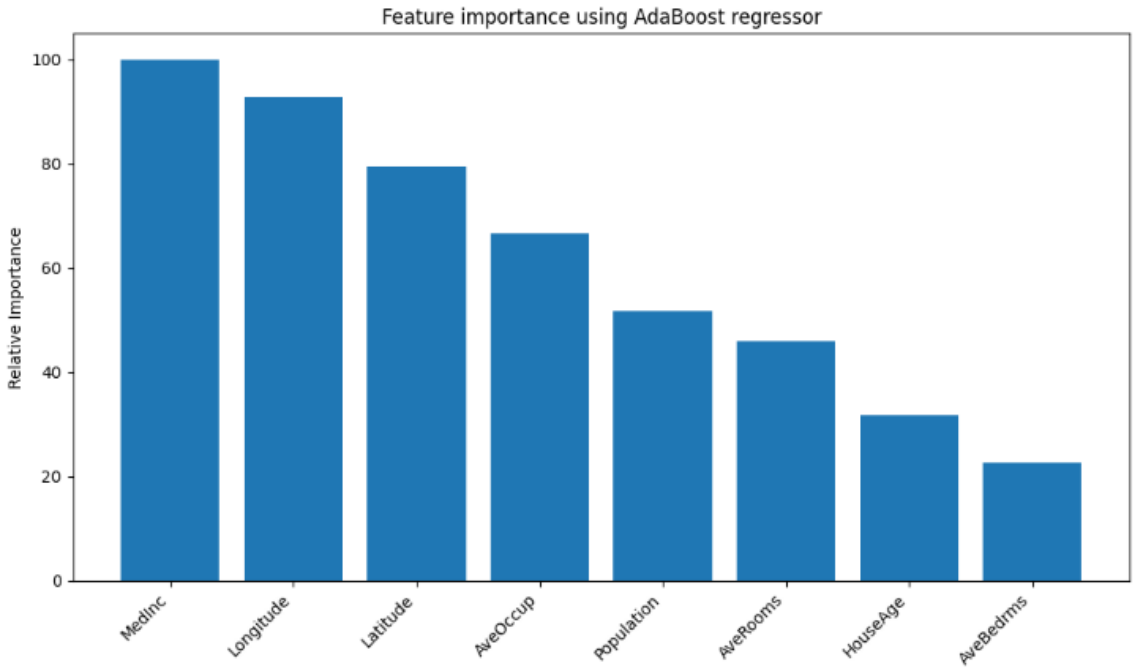

Рисунок 19 – Результат виконання програми

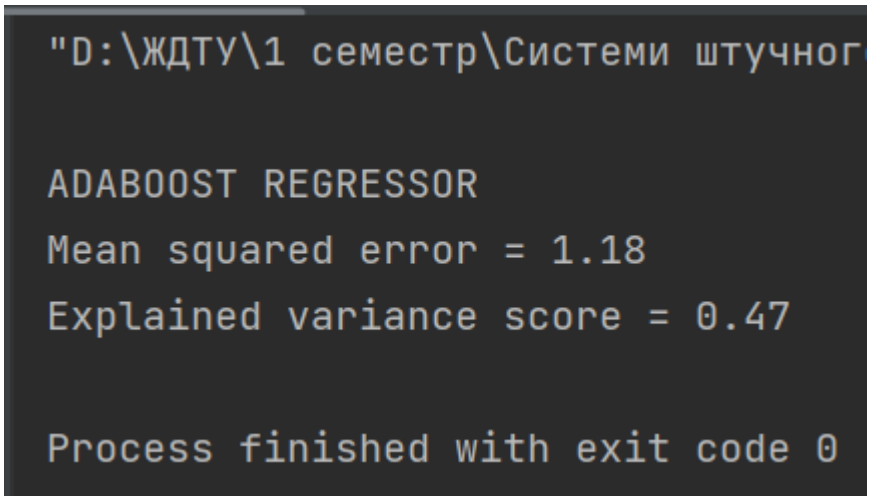

Рисунок 20 – Результат виконання програми

На основі проведеного аналізу, найважливішою ознакою для прогнозування цін на нерухомість є медіанний дохід (*MedInc*), оскільки його вплив найбільший серед усіх факторів. Це свідчить про сильну залежність вартості нерухомості від рівня доходів населення в регіоні. Географічне розташування, представлене довготою (*Longitude*) та широтою (*Latitude*), також має вагомий вплив, підтверджуючи важливість локації для визначення цін. Крім того, середня кількість мешканців у будинку (*AveOccup*) є суттєвим фактором, що може бути пов'язано із соціально-демографічними особливостями районів.

Середні значення для населення району (*Population*) та кількості кімнат у будинку (*AveRooms*) мають помірний вплив, що вказує на їх певну, але не критичну роль у формуванні цін. Менш важливими виявилися вік будинку (*HouseAge*) та середня кількість спалень (*AveBedrms*), що свідчить про їхній обмежений вплив на кінцевий результат. Таким чином, для побудови більш оптимізованої моделі варто зосередитися на ключових ознаках, таких як *MedInc*, *Longitude*, *Latitude* та *AveOccup*, тоді як менш значимі фактори можна враховувати опціонально або вилучати для спрощення аналізу.

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

Лістинг коду:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, mean_absolute_error
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.metrics import classification_report

# Завантаження вхідних даних
input_file = 'traffic_data.txt'
data = []
with open(input_file, 'r') as f:
    for line in f.readlines():
        items = line[:-1].split(',')
        data.append(items)

data = np.array(data)

# Перетворення рядкових даних на числові
label_encoder = []
X_encoded = np.empty(data.shape)
for i, item in enumerate(data[0]):
    if item.isdigit():
        X_encoded[:, i] = data[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(data[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

# Розбиття даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test size=0.25, random state=5)

# Регресор на основі гранично випадкових лісів
params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}
regressor = ExtraTreesRegressor(**params)
regressor.fit(X_train, y_train)

# Обчислення характеристик ефективності регресора на тестових даних
y_pred = regressor.predict(X_test)
print("Mean absolute error:", round(mean_absolute_error(y_test, y_pred), 2))

# Тестування кодування на одиночному прикладі
test_datapoint = ['Saturday', '10:20', 'Atlanta', 'no']
test_datapoint_encoded = [-1] * len(test_datapoint)
count = 0
for i, item in enumerate(test_datapoint):
    if item.isdigit():
        test_datapoint_encoded[i] = int(test_datapoint[i])
    else:
        test_datapoint_encoded[i] =
int(label_encoder[count].transform([test_datapoint[i]])[0])
        count = count + 1
test_datapoint_encoded = np.array(test_datapoint_encoded)

# Прогнозування результату для тестової точки даних
print("Predicted traffic:", int(regressor.predict([test_datapoint_encoded])[0]))
```

Результат виконання програми:

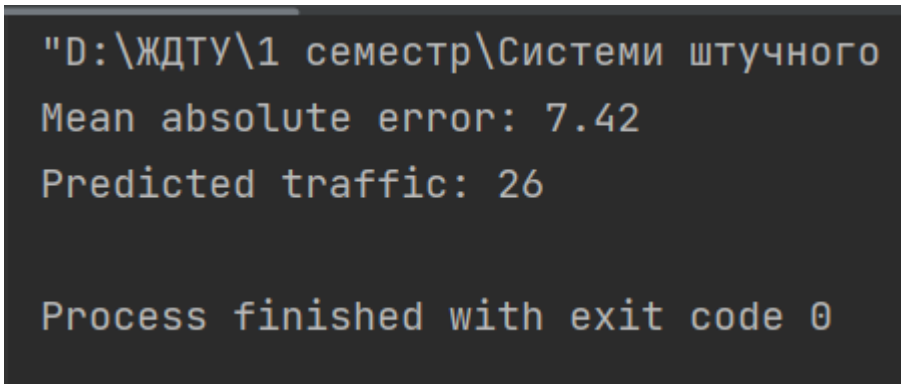

Рисунок 21 – Результат виконання програми

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