ЛАБОРАТОРНА РОБОТА № 5

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ

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

Варіант 1

Хід роботи:

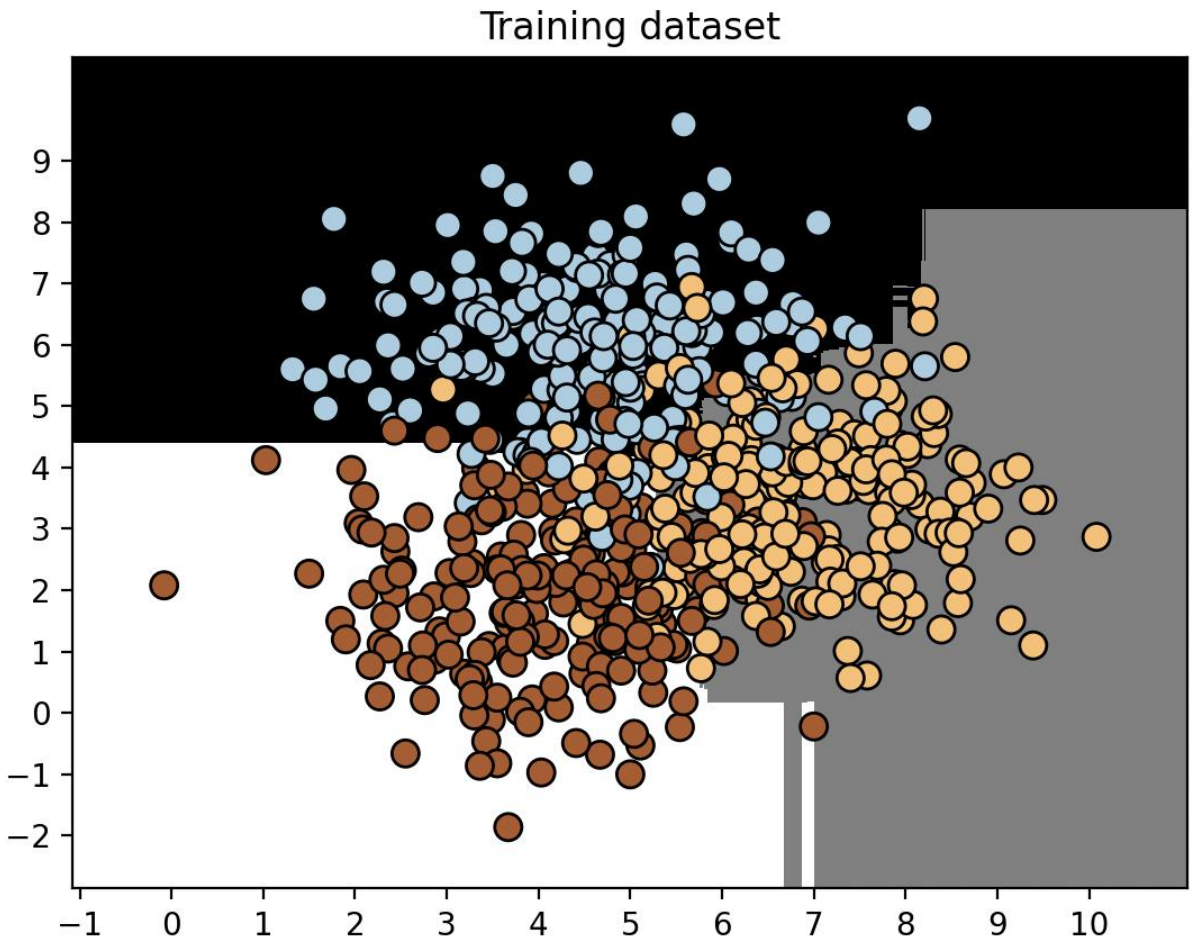
Завдання 2.1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

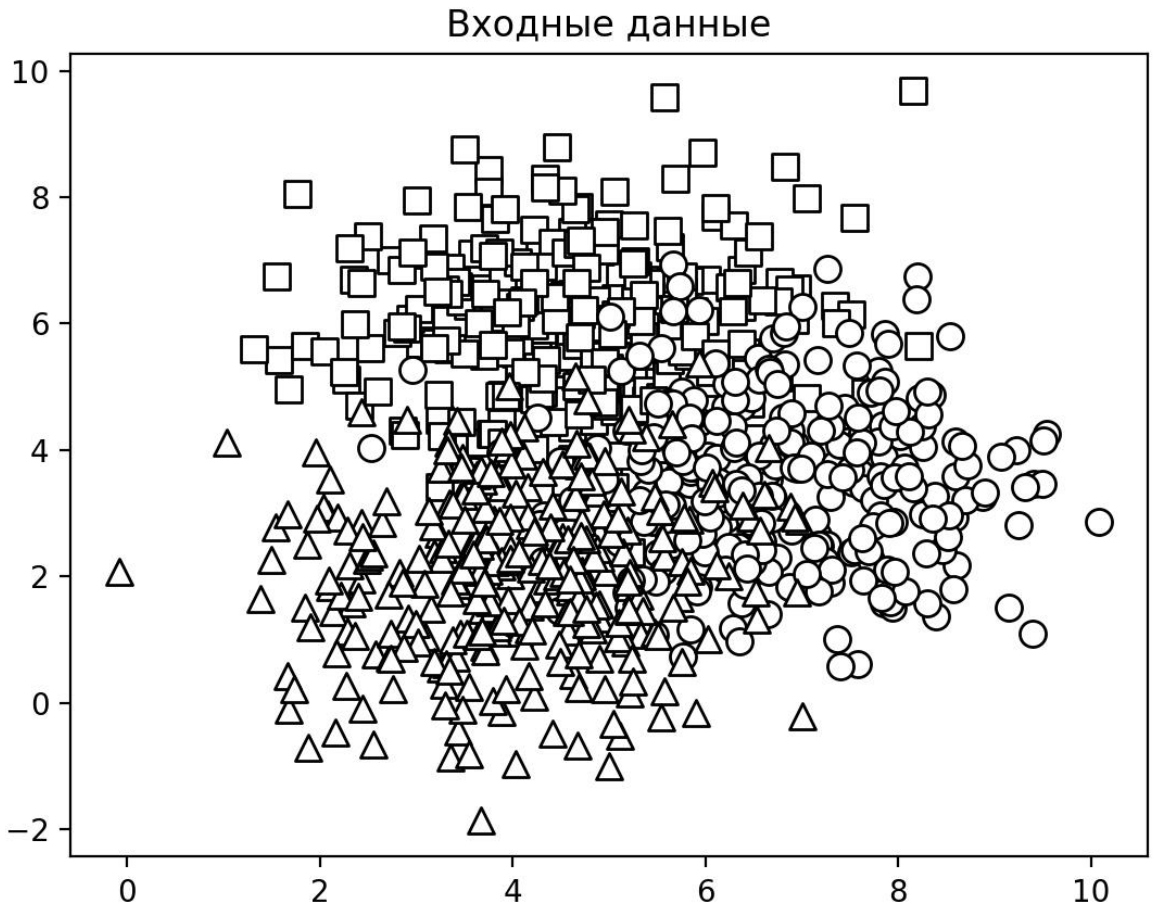
```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.metrics import classification_report
from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

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__':
    # Вилучення вхідних аргументів
    args = build_arg_parser().parse_args()
    classifier_type = args.classifier_type
```

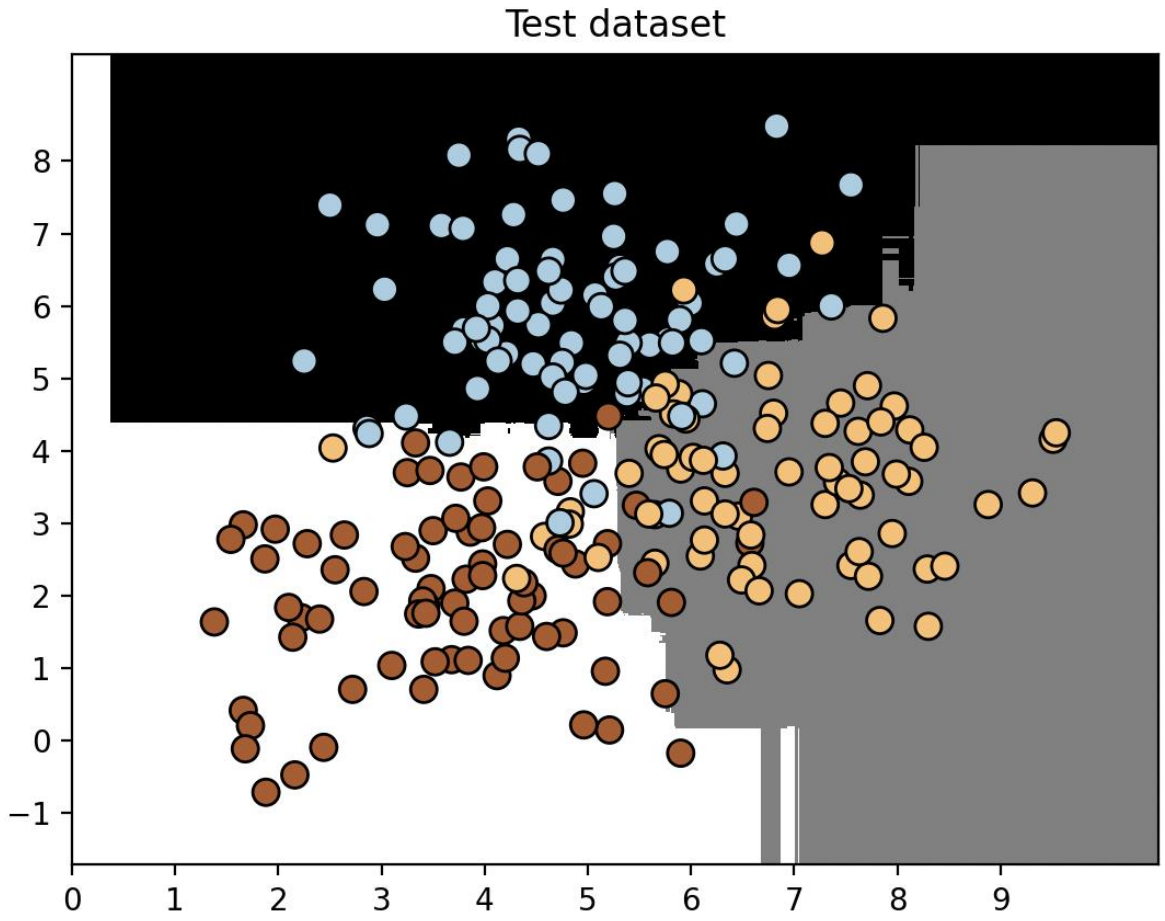
```
    # Завантаження вхідних даних
    input_file = 'data_random_forests.txt'
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]

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



    # Візуалізація вхідних даних
    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('Входные данные')

    # Розіб'ємо дані 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 ансамблевого навчання
    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, 'Training dataset')

    y_test_pred = classifier.predict(X_test)
    visualize_classifier(classifier, X_test, y_test, 'Test dataset')

    # Перевірка роботи класифікатора
    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")
```

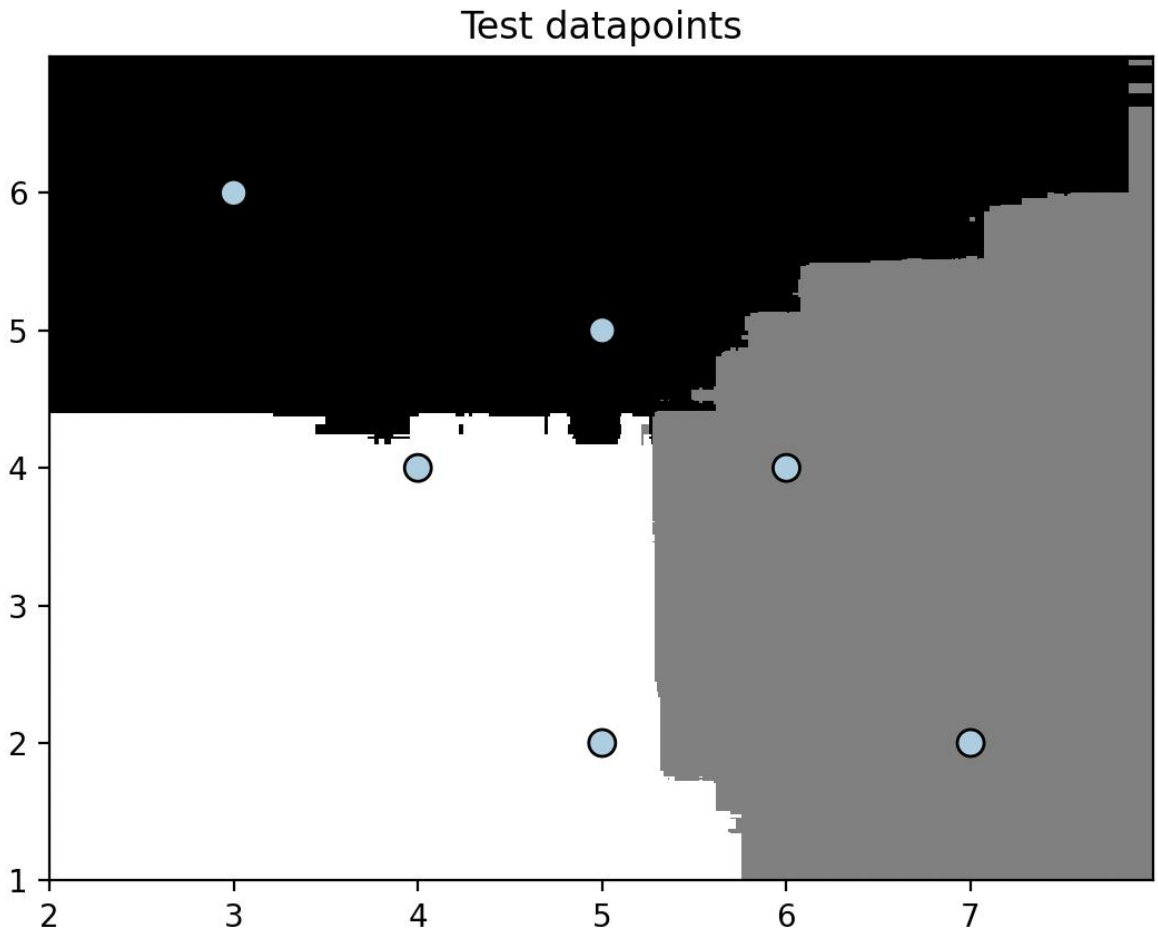
```
    print("#" * 40)
    print("\nClassifier performance on test dataset\n")
    print(classification_report(y_test, y_test_pred,
target_names=class_names))
    print("#" * 40 + "\n")
    test_datapoints = np.array([[5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5,
2]])

    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)
        print('Probability:', np.max(probabilities))

    # Візуалізація точок даних
    visualize_classifier(classifier, test_datapoints, [0] *
len(test_datapoints), 'Test datapoints')
    plt.show()
```

Random Forest Classifier



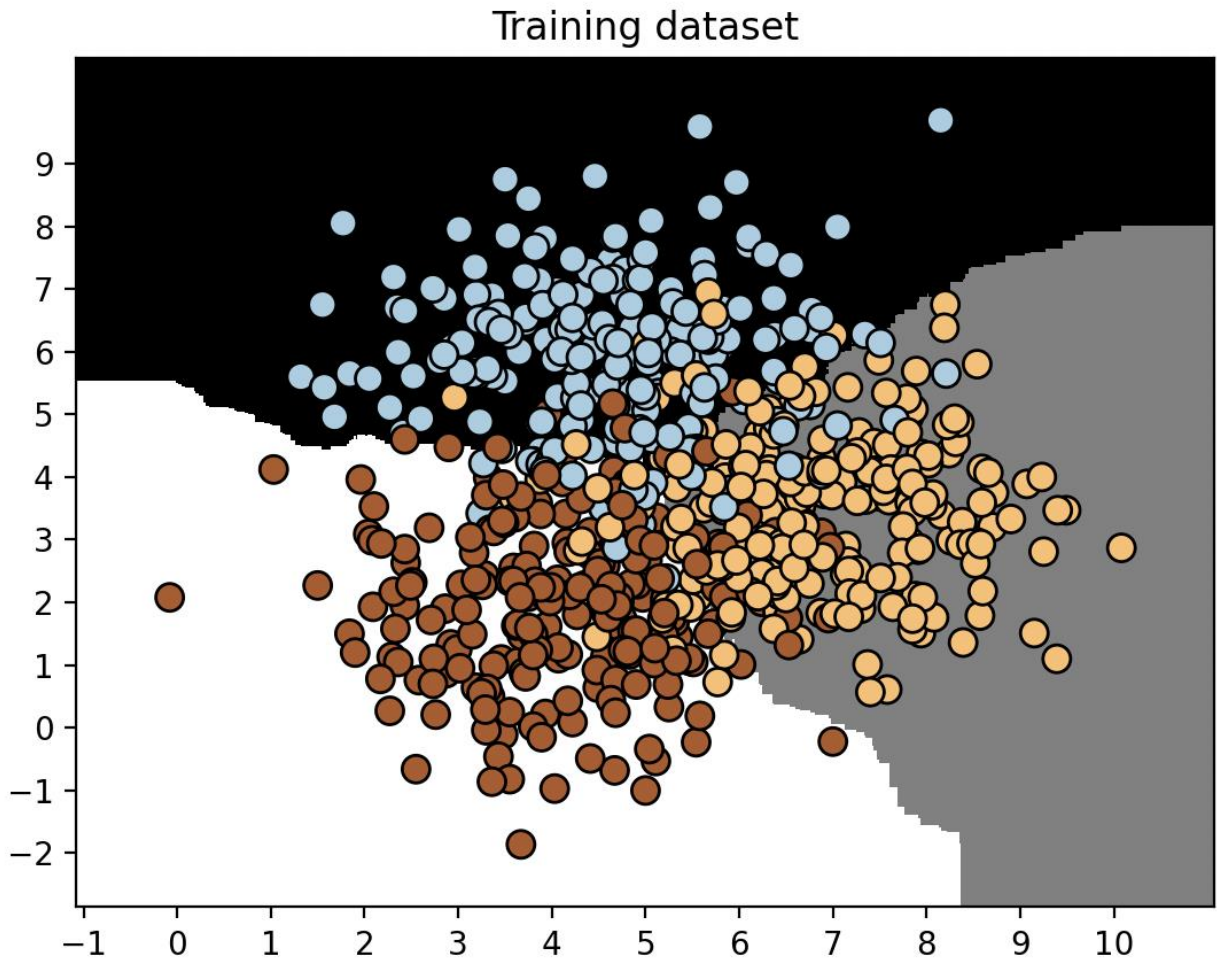

Входные данные
10
8
6
4
2
0
−2
0
2
4
6
8
10

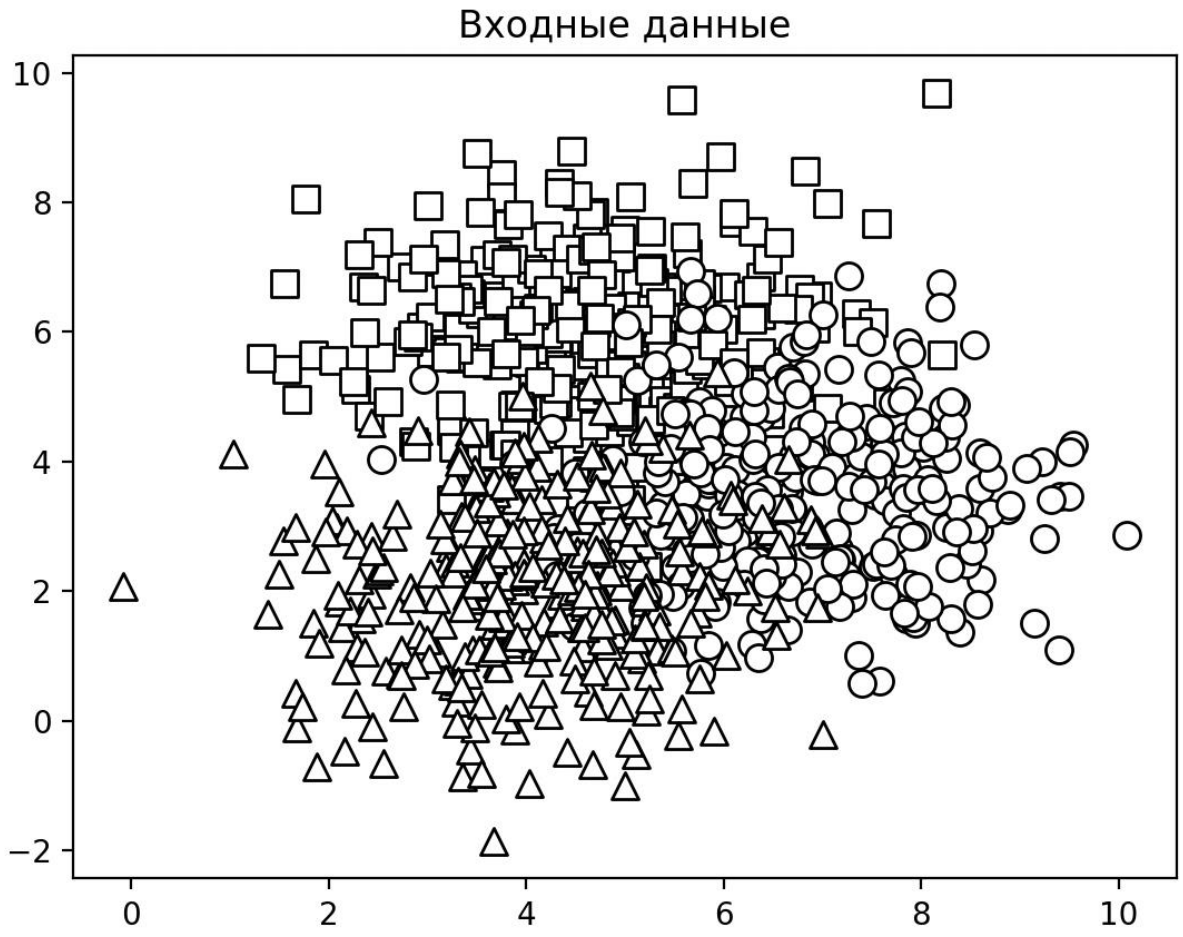

Test dataset
8
7
6
5
4
3
2
1
0
−1
0
1
2
3
4
5
6
7
8
9

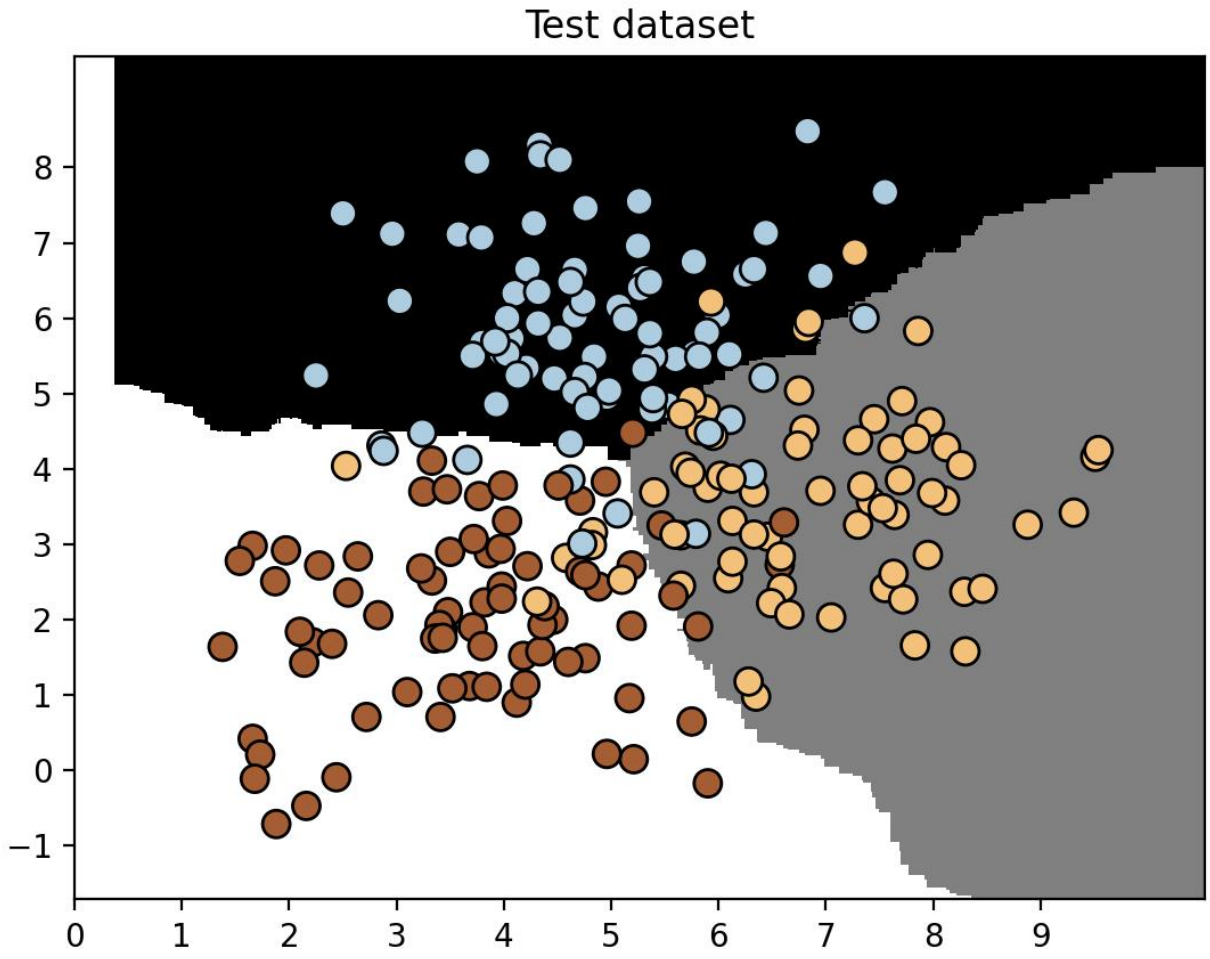

Test datapoints
6
5
4
3
2
1
2
3
4
5
6
7

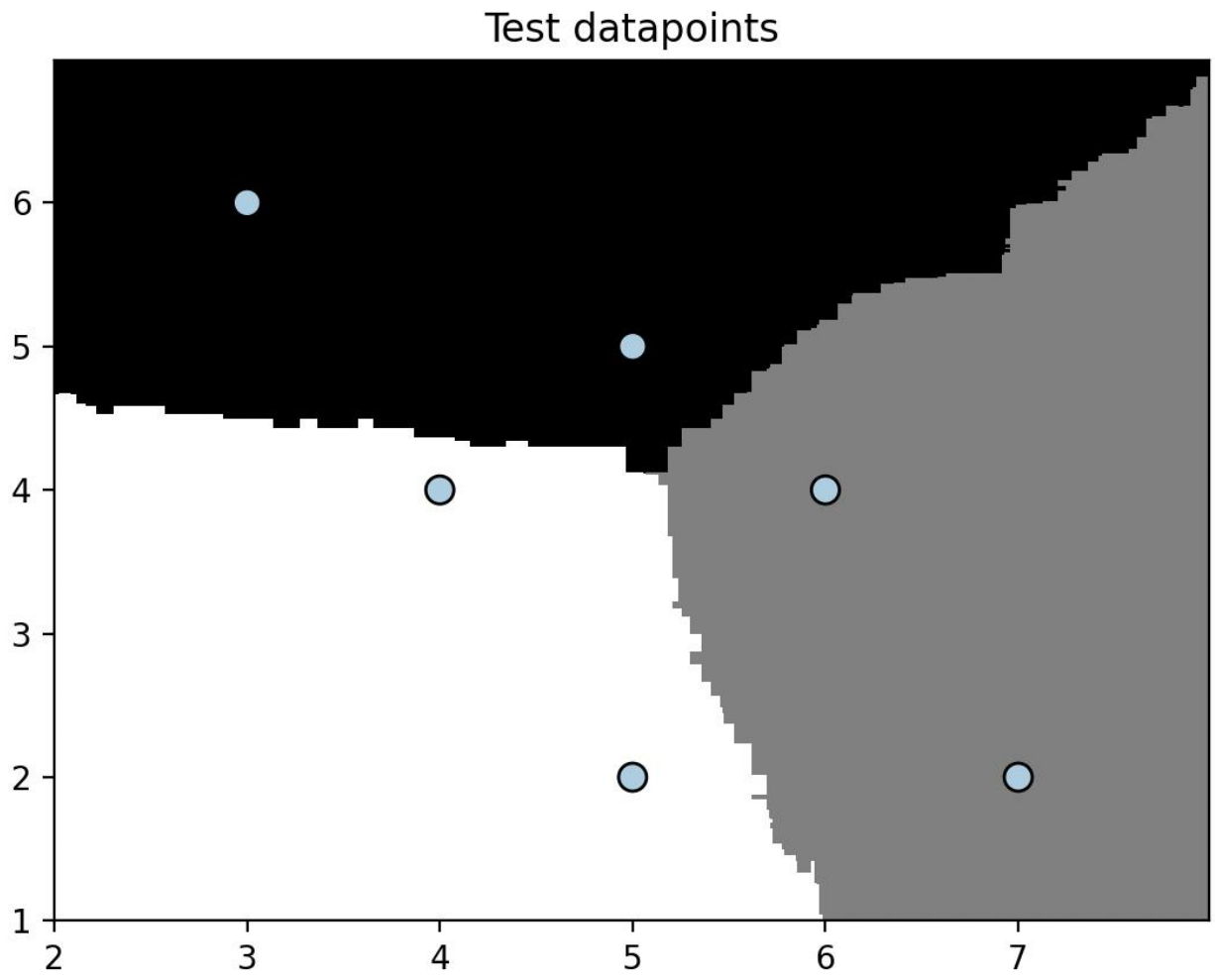
```
2024-12-04 00:22:04.551 Python[57859:12458048] +[IMKClient subclass]: chose IMKClient_Legacy
2024-12-04 00:22:04.552 Python[57859:12458048] +[IMKInputSession subclass]: chose IMKInputSession_Legacy

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

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

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


Confidence measure:

Datapoint: [5 5]
Predicted class: Class-0
Probability: 0.814275318675482

Datapoint: [3 6]
Predicted class: Class-0
Probability: 0.9357445782050678

Datapoint: [6 4]
Predicted class: Class-1
Probability: 0.7451078021772837

Datapoint: [7 2]
Predicted class: Class-1
Probability: 0.7066022643054627

Datapoint: [4 4]
Predicted class: Class-2
Probability: 0.6388176473387874

Datapoint: [5 2]
Predicted class: Class-2
Probability: 0.8528526682816628
```

Extra Trees Classifier:


Training dataset
9
8
7
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6
7
8
9
10


Входные данные
10
8
6
4
2
0
−2
0
2
4
6
8
10


Test dataset
8
7
6
5
4
3
2
1
0
−1
0
1
2
3
4
5
6
7
8
9


Test datapoints
6
5
4
3
2
1
2
3
4
5
6
7

```
2024-12-04 00:23:54.792 Python[58679:12462717] +[IMKClient subclass]: chose IMKClient_Legacy
2024-12-04 00:23:54.792 Python[58679:12462717] +[IMKInputSession subclass]: chose IMKInputSession_Legacy

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

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

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


Confidence measure:

Datapoint: [5 5]
Predicted class: Class-0
Probability: 0.4890441872546998

Datapoint: [3 6]
Predicted class: Class-0
Probability: 0.6670738316255893

Datapoint: [6 4]
Predicted class: Class-1
Probability: 0.49535143822679545

Datapoint: [7 2]
Predicted class: Class-1
Probability: 0.6246676978884234

Datapoint: [4 4]
Predicted class: Class-2
Probability: 0.4512103944624855

Datapoint: [5 2]
Predicted class: Class-2
Probability: 0.5256798857566317
```

Висновок:

Алгоритм RandomForestClassifier формує підвибірки даних за допомогою бутстреп-методу, на основі яких будує дерево рішень. Остаточний клас визначається голосуванням, при якому обирається варіант з найбільшою кількістю голосів. Цей метод характеризується високою точністю, стійкістю до шумових даних і низькою ймовірністю перенавчання.

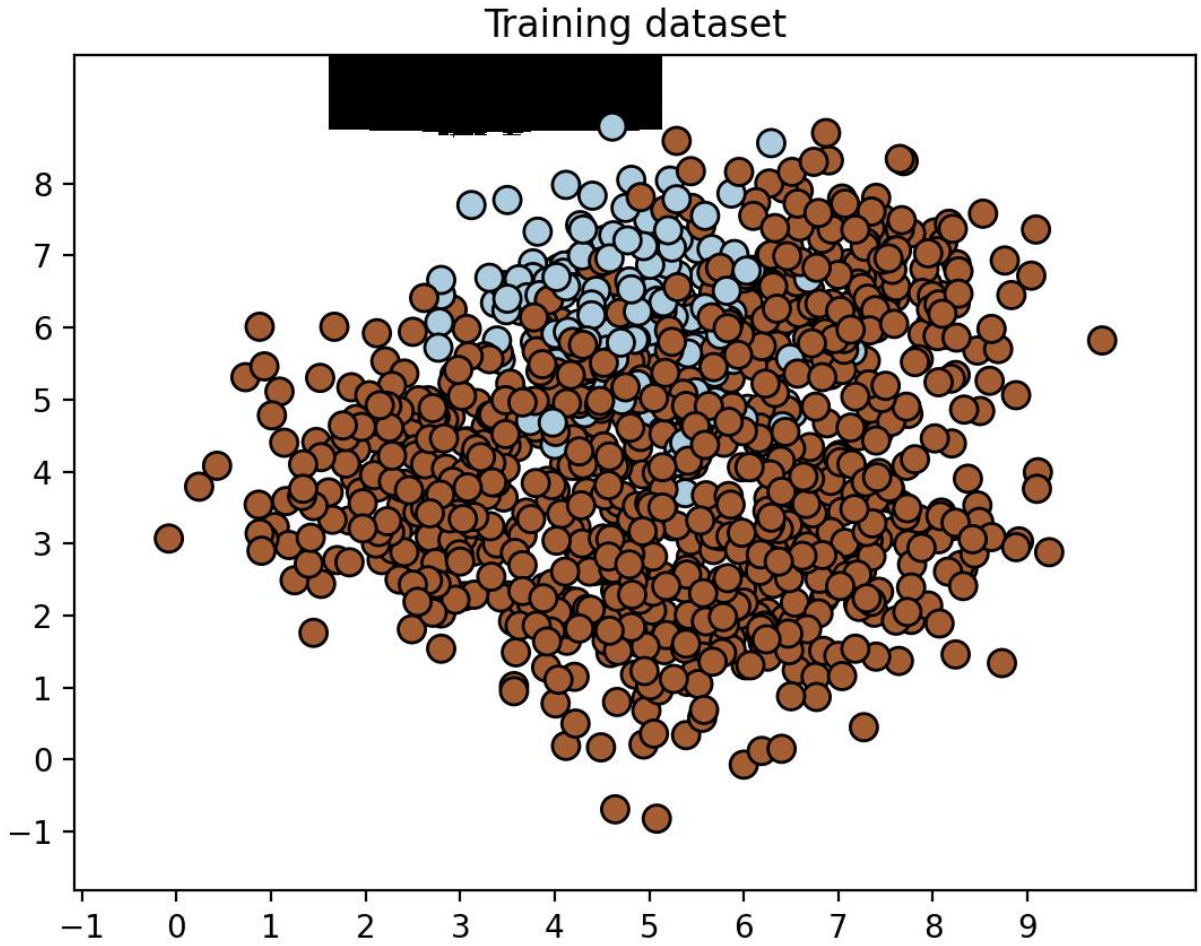
Алгоритм ExtraTreesClassifier, у свою чергу, працює за подібним принципом, однак використовує всі доступні ознаки і випадковим чином обирає пороги

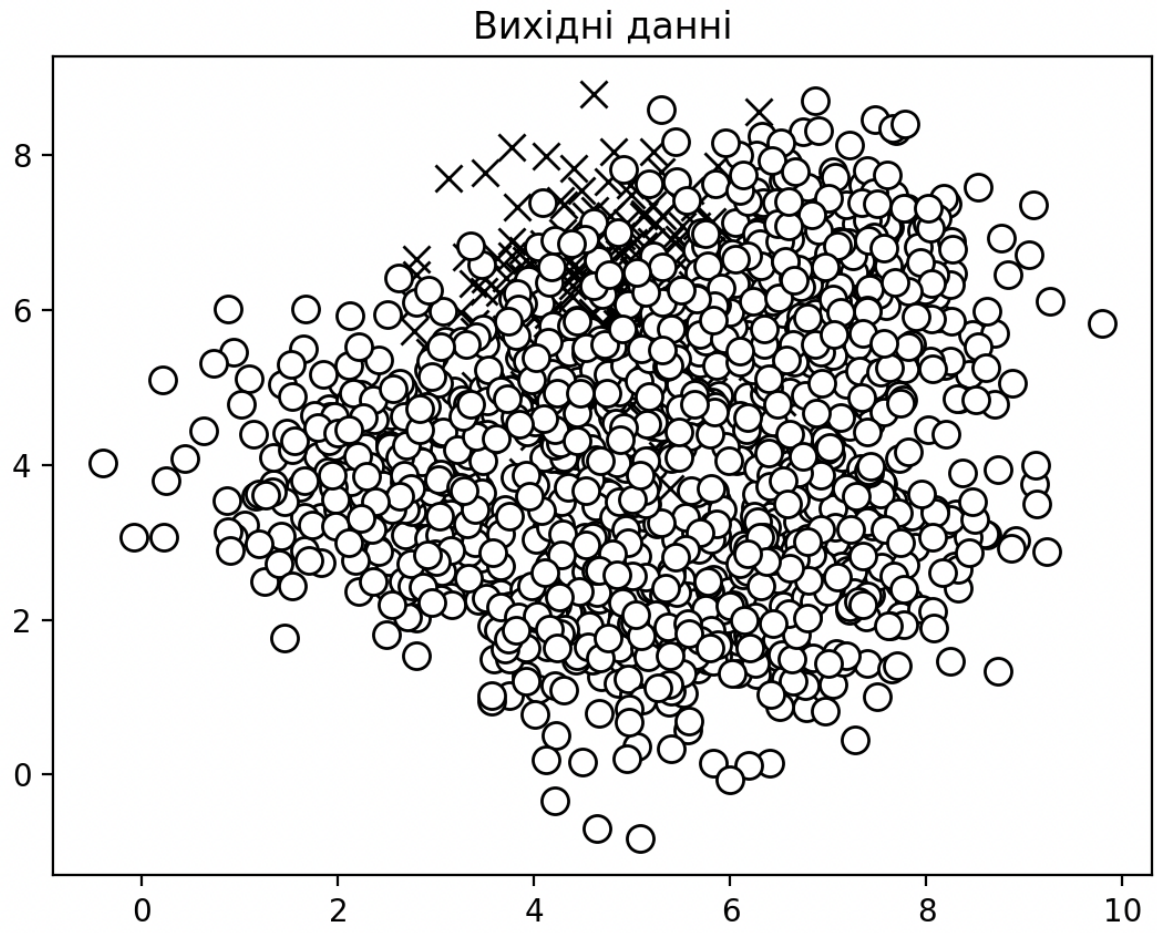
розділення. Це дозволяє досягти високої продуктивності в задачах класифікації з великою кількістю вхідних характеристик.

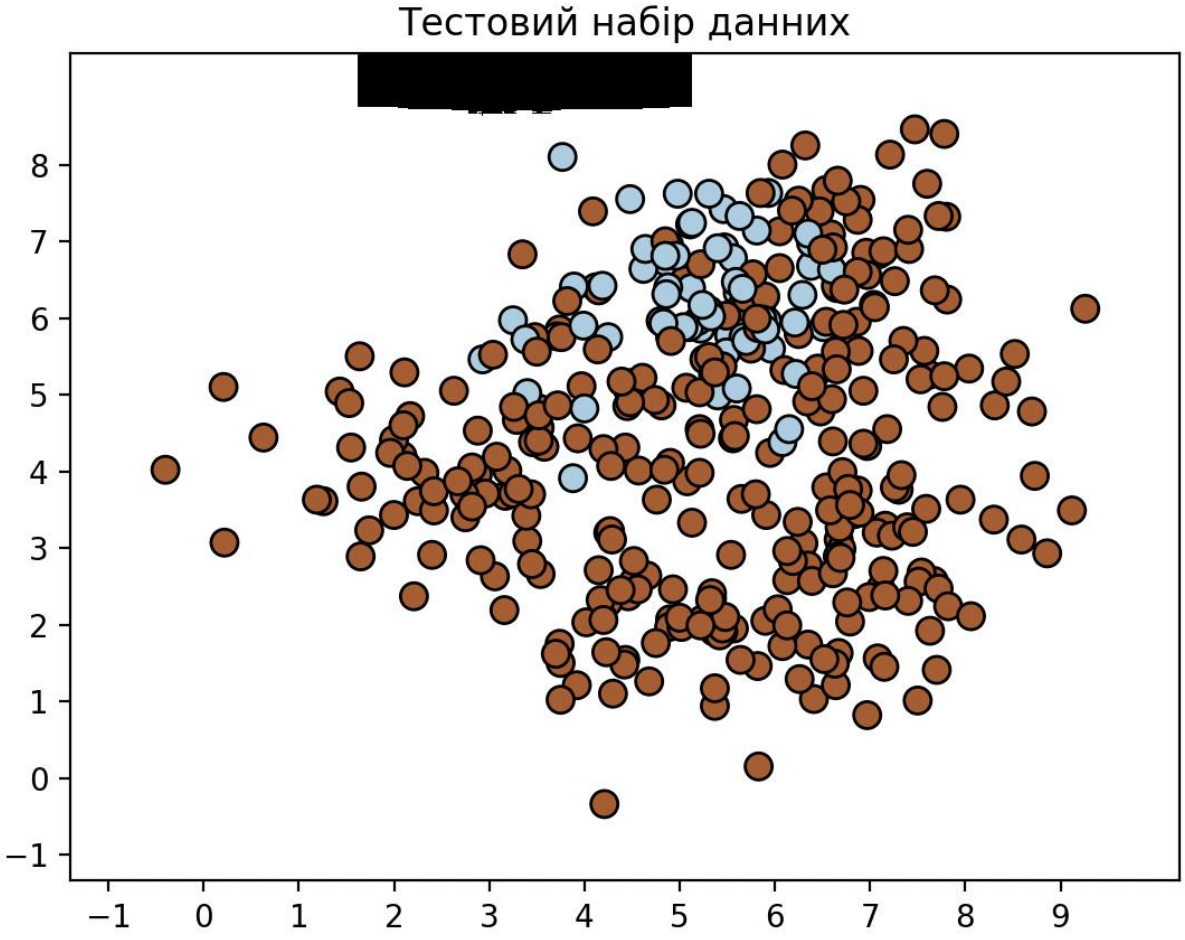
Завдання 2.2. Обробка дисбалансу класів

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.ensemble import ExtraTreesClassifier
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('Вихідні данні')

# 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, 'Training dataset')

y_test_pred = classifier.predict(X_test)
```

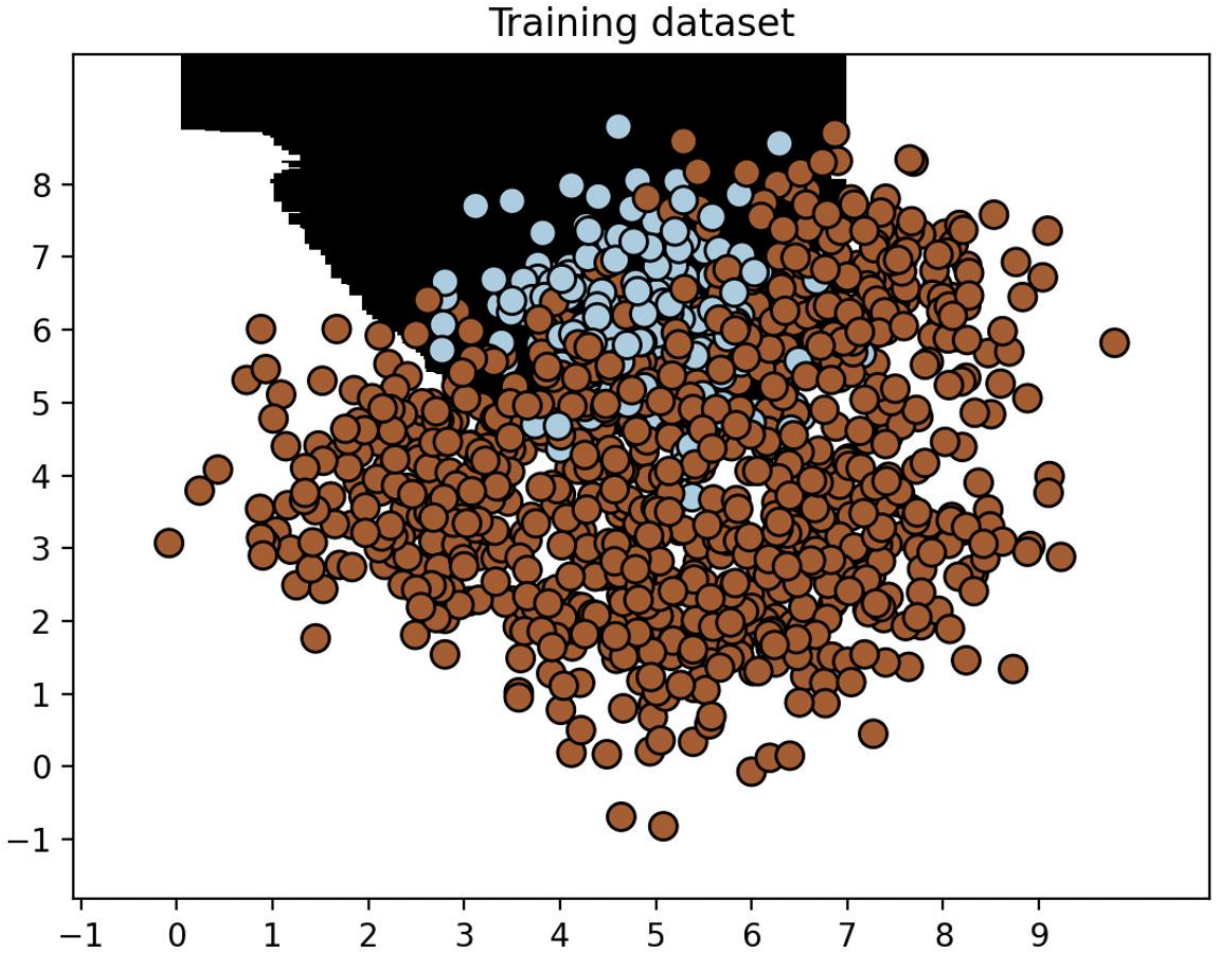
```
visualize_classifier(classifier, X_test, y_test, 'Тестовий набір данних')

# 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()
```

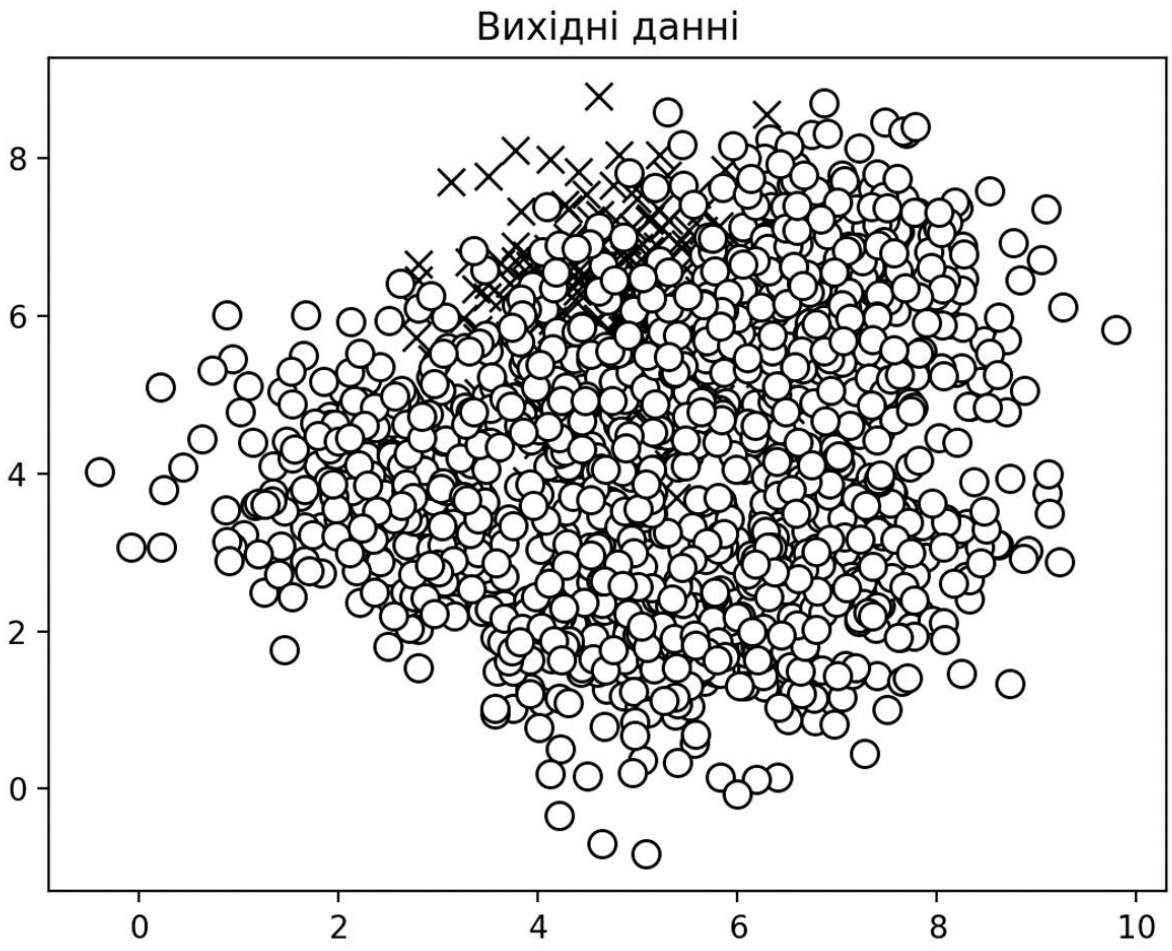


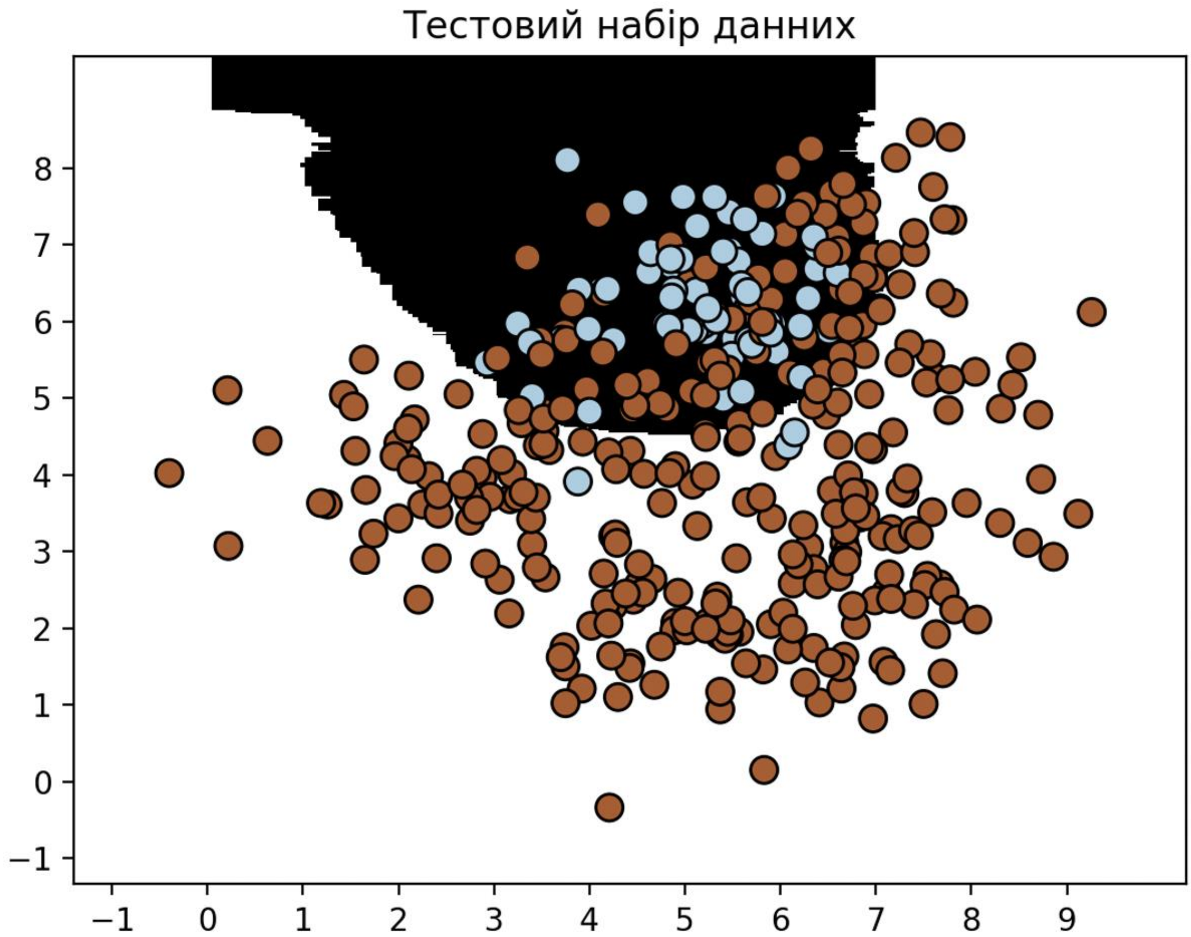

Вихідні данні
0
2
4
6
8
0
2
4
6
8
10

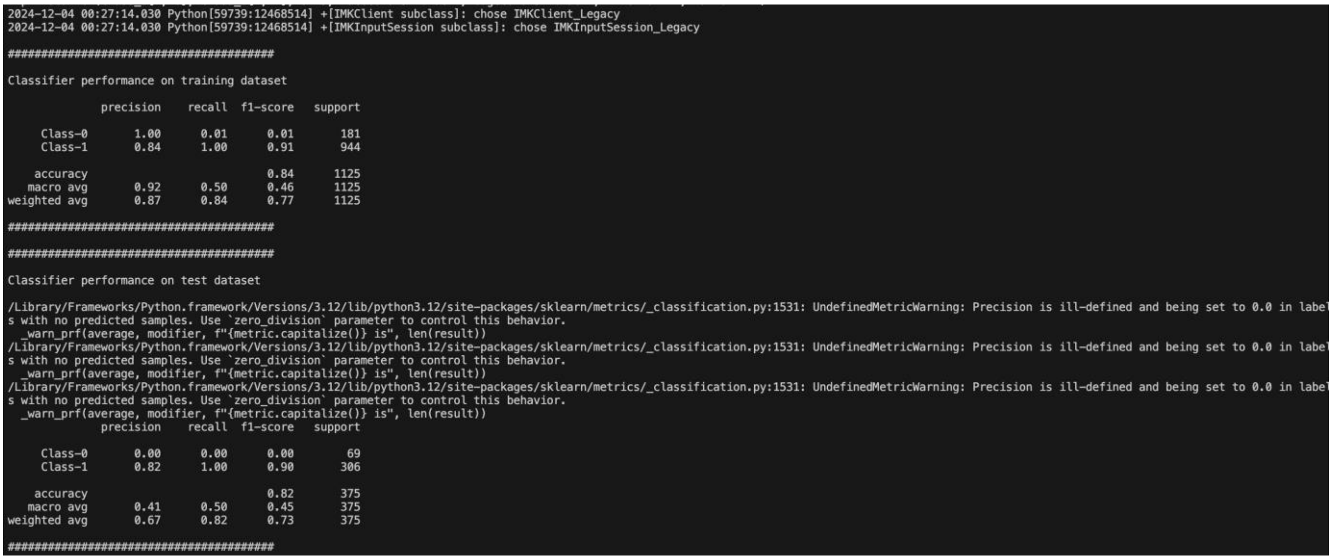


Balance:


Training dataset
8
7
6
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5
6
7
8
9


Вихідні данні
8
6
4
2
0
0
2
4
6
8
10





```
2024-12-04 00:27:14.030 Python[59739:12468514] +[IMKClient subclass]: chose IMKClient_Legacy
2024-12-04 00:27:14.030 Python[59739:12468514] +[IMKInputSession subclass]: chose IMKInputSession_Legacy

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

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

/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in label
s with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in label
s with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in label
s with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
              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

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

Висновок: Якщо для параметра `class_weight` у алгоритмі ExtraTreesClassifier встановити значення `balanced`, ваги класів будуть автоматично розраховані на основі їхньої частоти у навчальній вибірці. Це допомагає зменшити вплив дисбалансу в даних, надаючи більше значення менш чисельним класам, що сприяє підвищенню точності класифікації для них.

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

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

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]}]
metrics = ['precision_weighted', 'recall_weighted']

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)

    # Використання cv_results_ для виведення результатів
    print("\nGrid scores for the parameter grid:")
    for mean_score, params in zip(classifier.cv_results_['mean_test_score'],
classifier.cv_results_['params']):
        print(params, '-->', round(mean_score, 3))

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

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

```
#### 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}

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

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

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
```

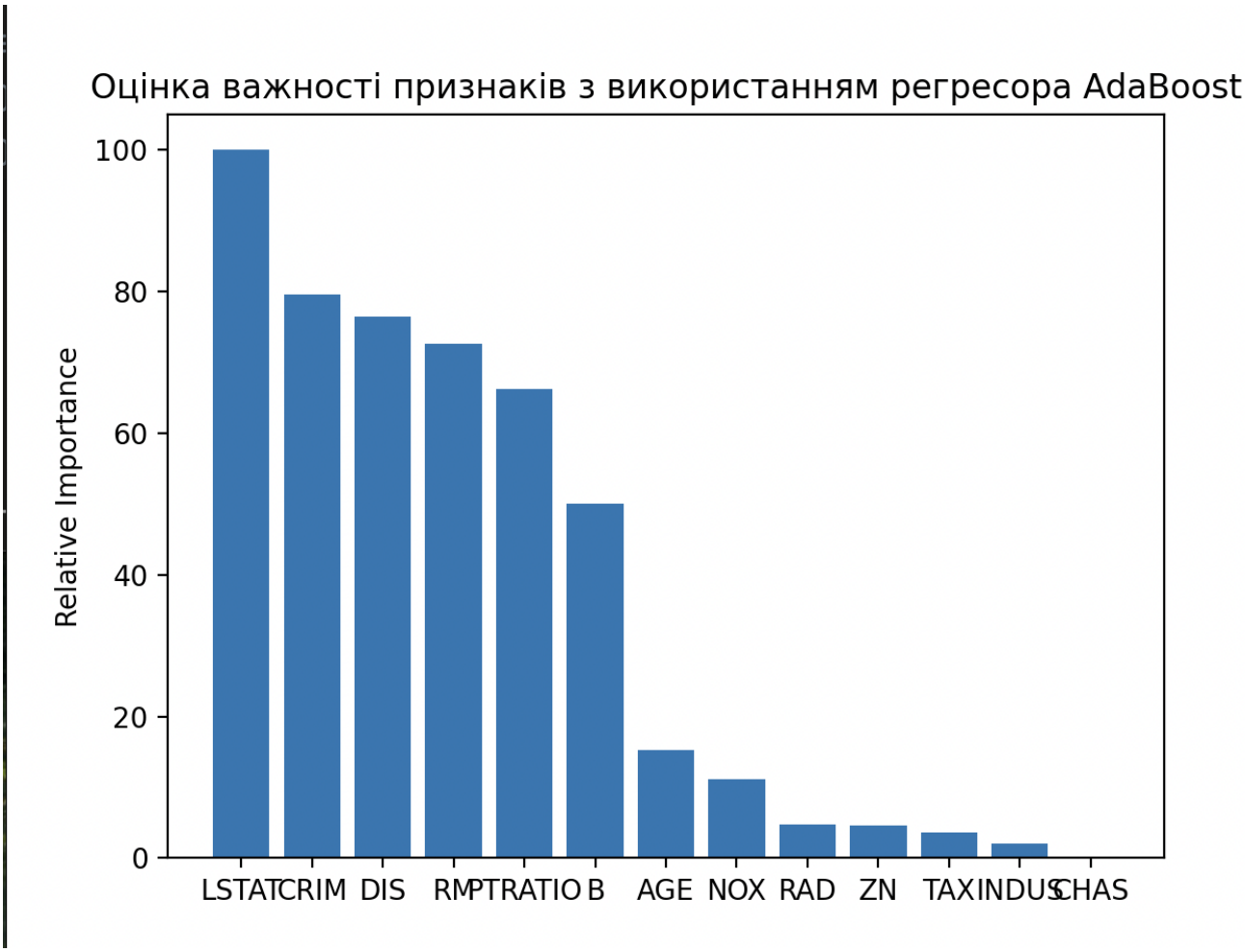
Висновок: GridSearchCV використовується для пошуку оптимальних параметрів моделі шляхом перебору всіх можливих комбінацій із заданої сітки та оцінки їхньої ефективності за допомогою крос-валідації. У наведеному прикладі цей метод оптимізує параметри `n_estimators` і `max_depth` для ExtraTreesClassifier. Отримані результати відображають середні оцінки для кожної комбінації, що дозволяє обрати найкращі значення параметрів. Це сприяє покращенню точності моделі, навіть за наявності складної структури даних або дисбалансу між класами.

Завдання 2.4. Обчислення відносної важливості ознак

```python
import numpy as np
import pandas as pd
import ssl
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error, explained_variance_score
from sklearn.utils import shuffle
import matplotlib.pyplot as plt

# Вимкнення перевірки SSL (тільки для локального використання)
ssl._create_default_https_context = ssl._create_unverified_context

# Завантаження даних
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
housing_data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

# Перетворення цільових змінних
label_encoder = preprocessing.LabelEncoder()
y = label_encoder.fit_transform(target)

# Перемішування даних
X, y = shuffle(housing_data, y, 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 = AdaBoostClassifier(DecisionTreeClassifier(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_

# Приклад імен ознак (можна задати вручну, якщо дані не мають назв)
feature_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']

# 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
plt.figure()
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, [feature_names[i] for i in index_sorted])
plt.ylabel('Relative Importance')
plt.title('Оцінка важності признаків з використанням регресора AdaBoost')
plt.show()
```




```
ADABOOST REGRESSOR
Mean squared error = 1539.78
Explained variance score = 0.63
2024-12-04 00:29:58.228 Python[61363:12477473] +[IMKClient subclass]: chose IMKClient_Legacy
2024-12-04 00:29:58.229 Python[61363:12477473] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
```

Висновок: Діаграма ілюструє важливість ознак у моделі регресора **AdaBoost** для прогнозування вартості житла. Найбільший вплив мають такі характеристики, як

LSTAT (відсоток населення з низьким соціально-економічним статусом), **CRIM** (рівень злочинності) та **DIS** (віддаленість від робочих центрів). Значний внесок також роблять **RM** (середня кількість кімнат у будинку) і **PTRATIO** (співвідношення учнів до вчителів у школах району). У той же час ознаки, такі як **TAX** (податкове навантаження на нерухомість), **INDUS** (частка промислових зон у місті) та **CHAS** (близькість до річки Чарльз), мають мінімальний вплив, що свідчить про їхню незначущість для цієї моделі. Такий аналіз дозволяє зосередитися на ключових змінних, щоб покращити точність прогнозів.

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

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

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 += 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]))
```

```
Mean absolute error: 7.42
Predicted traffic: 26
```

Посилання на Github:

https://github.com/Vladislaw2533/SHI_Barabash_Vlad_IPZ_21_3

Висновки: використав спеціалізовані бібліотеки та мову програмування Python, та дослідив методи ансамблів у машинному навчанні.