

ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ ТА СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

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

Завдання №1: Створення класифікаторів на основі випадкових та гранично випадкових лісів.

```
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

# Парсер аргументів
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])
```

					ДУ «Житомирська політехніка».22.121.14.000 – Лр4			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Розроб.		Сірач А.С.					1	30
Перевір.		Філіпов В.О.				ФІКТ Гр. ІПЗ-19-3[2]		
Керівник								
Н. контр.								
Зав. каф.								

```

# 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('Input data')

# Розбивка 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 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")

# Обчислення параметрів довірливості
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)

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

```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

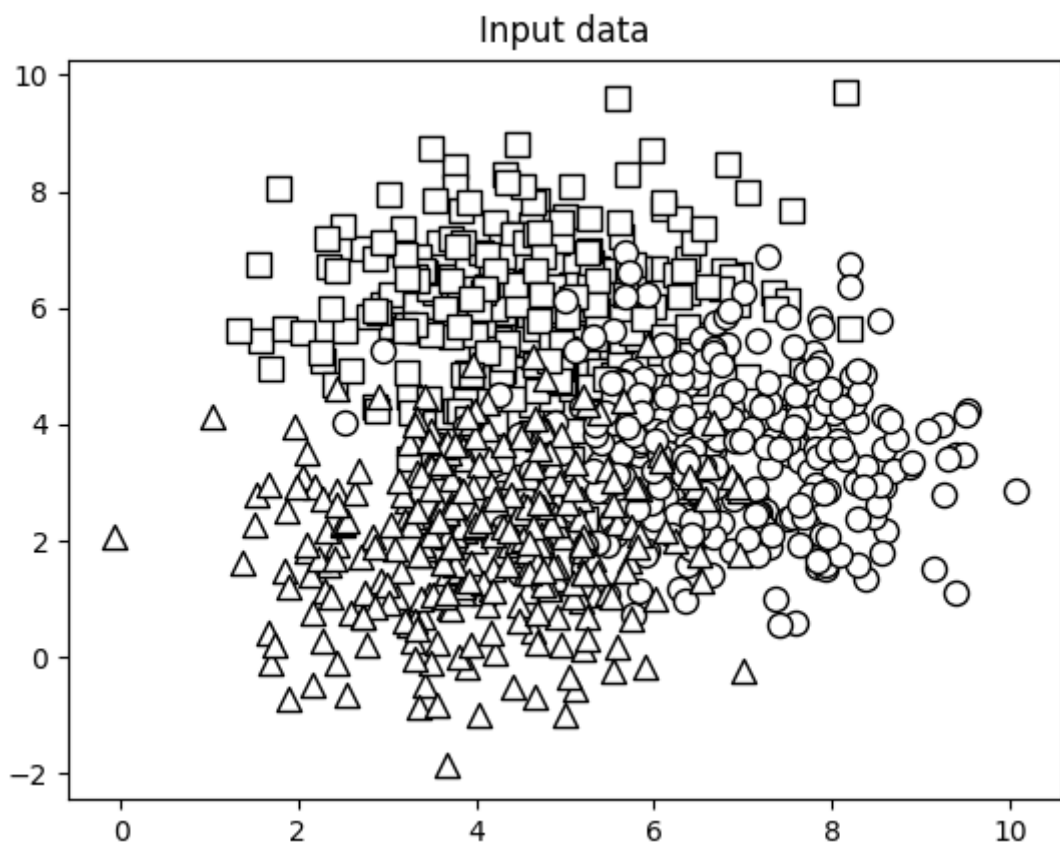


Рис. 1. Результат виконання програми

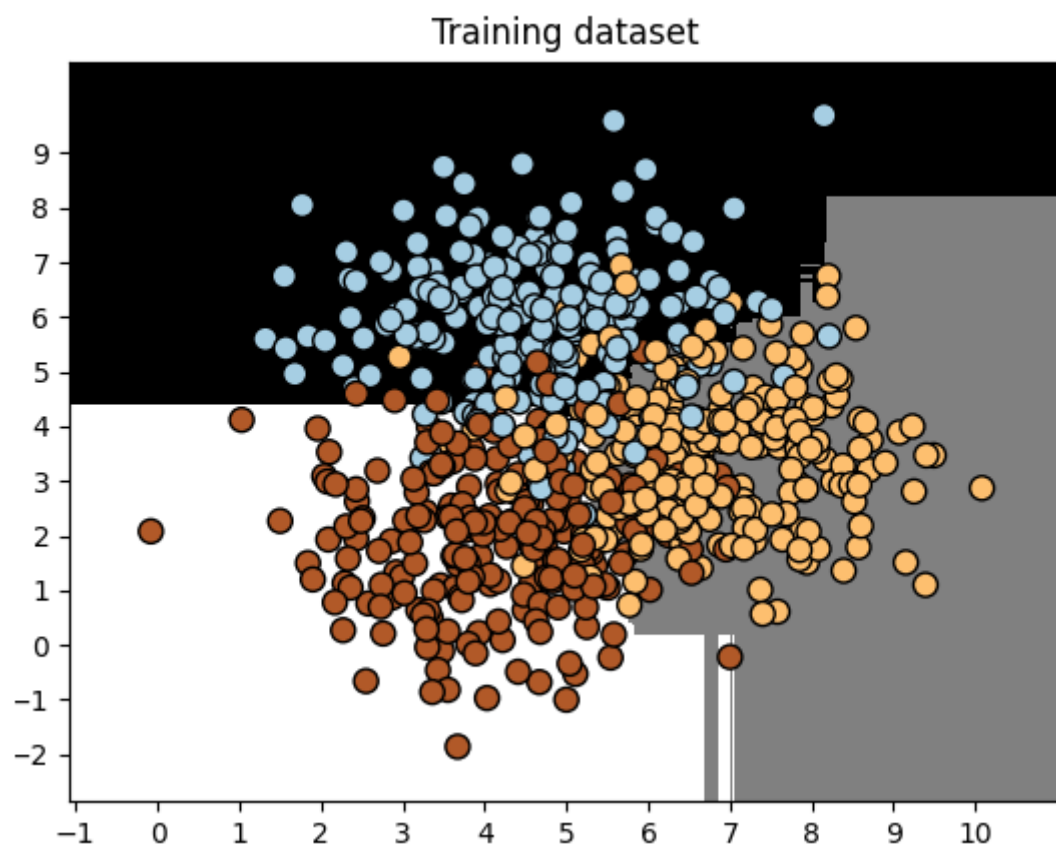


Рис. 2. Результат виконання програми

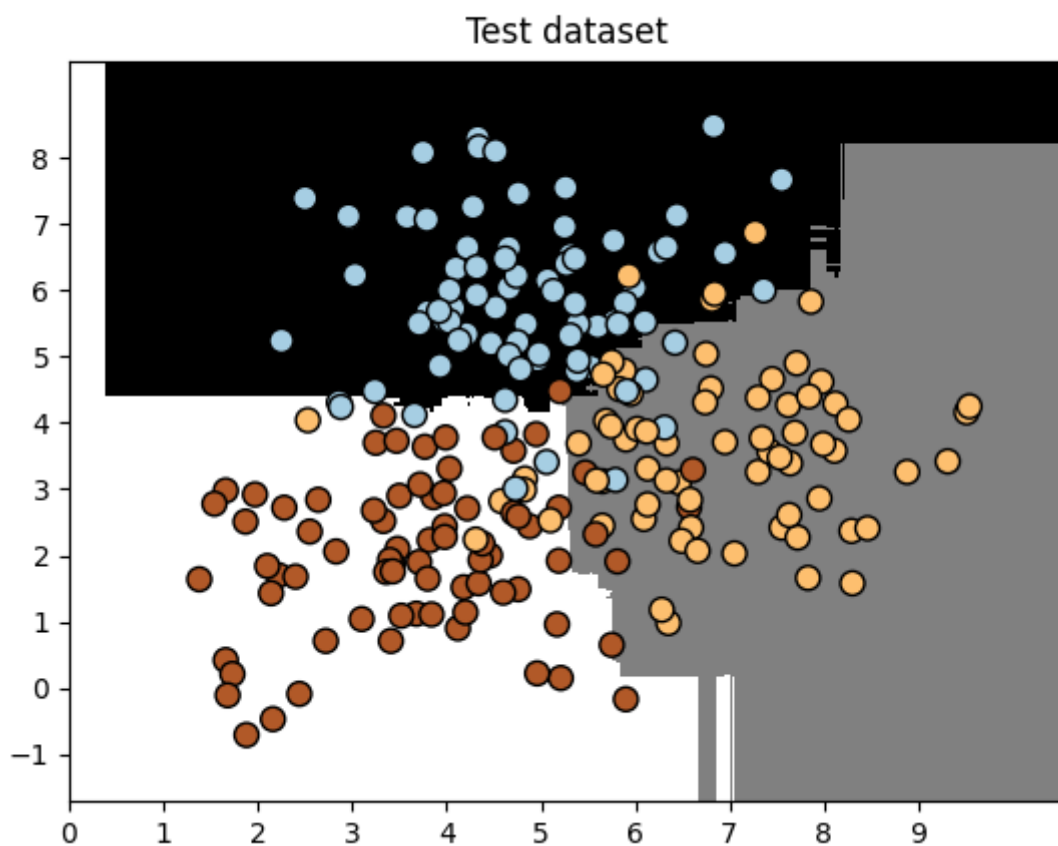


Рис. 3. Результат виконання програми

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. 4. Результат виконання програми

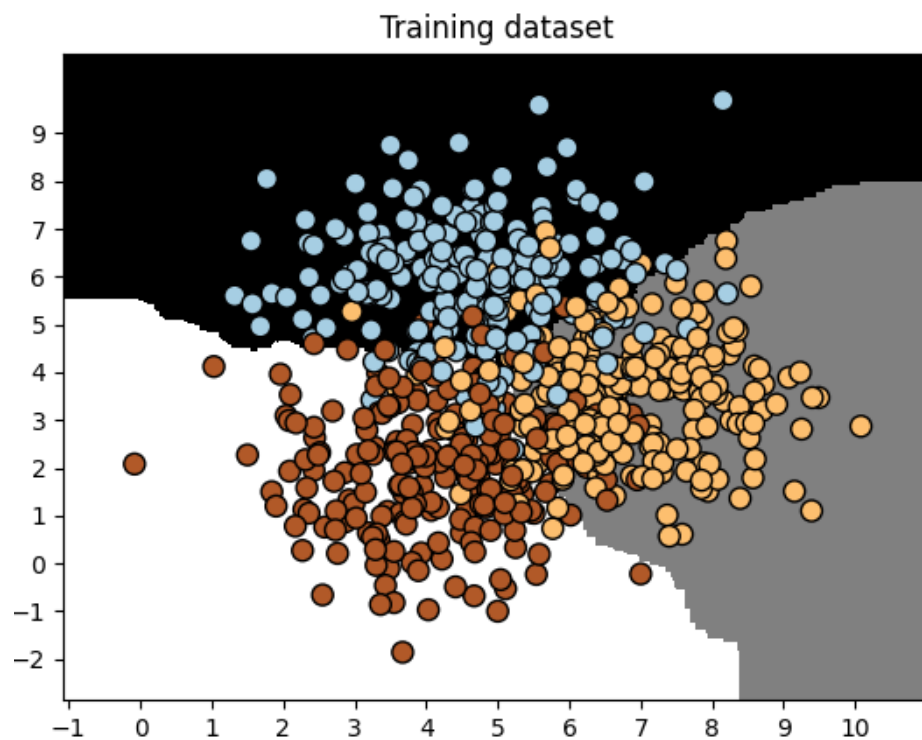


Рис. 5. Результат виконання програми

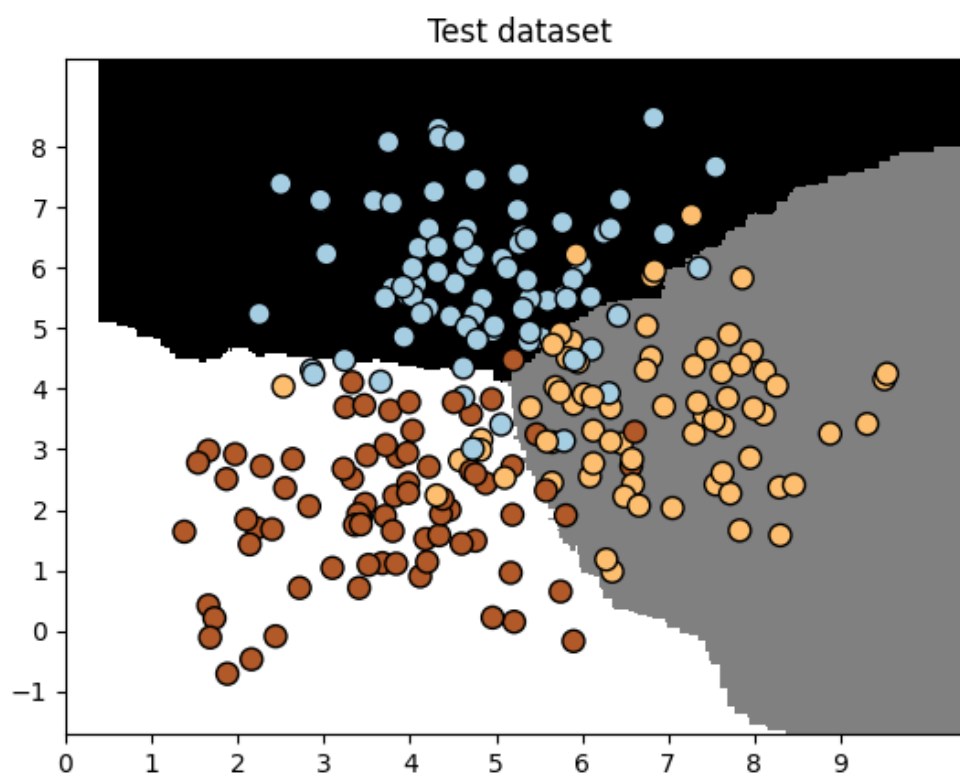


Рис. 6. Результат виконання програми

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Рис. 7. Результат виконання програми

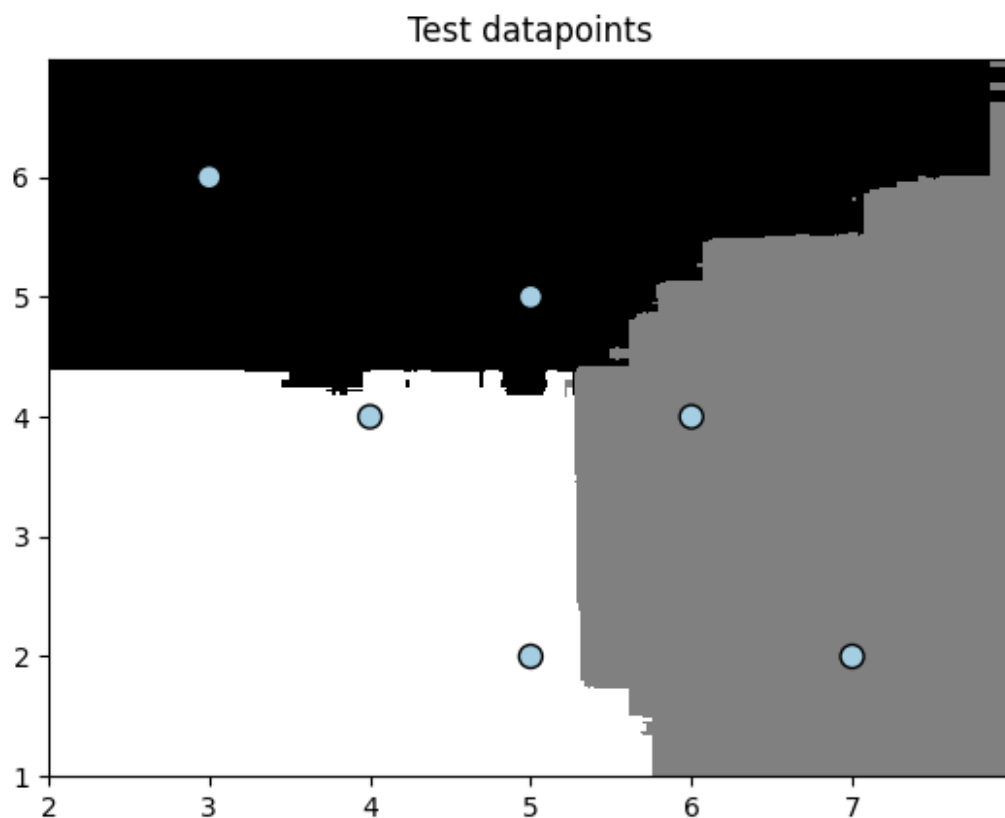


Рис. 8. Результат виконання програми

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

Рис. 9. Результат виконання програми

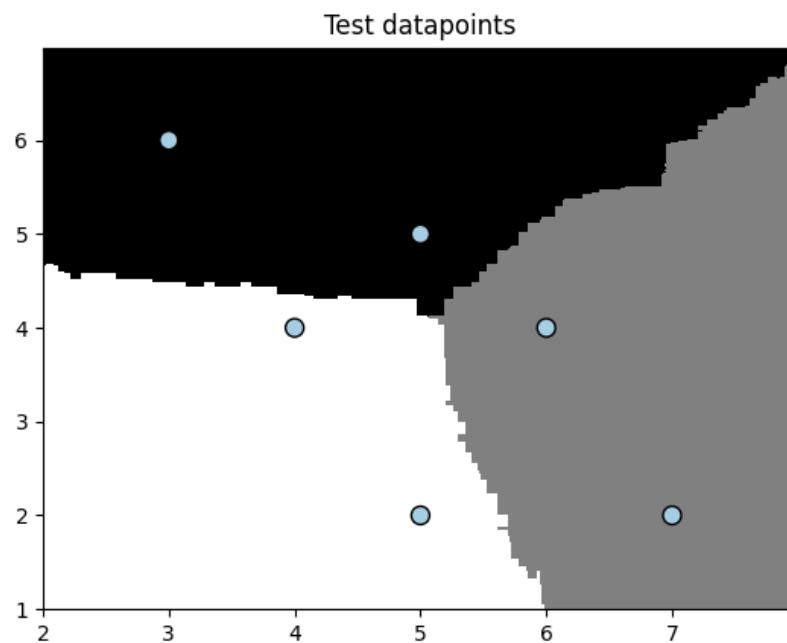


Рис. 10. Результат виконання програми

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Рис. 11. Результат виконання програми

Під час використання **-erf** було отримано більш точні виміри. Це обумовлено тим, що в процесі навчання гранично випадкові ліси мають більше можливостей для вибору оптимальних дерев рішень, тому, як правило, вони забезпечують отримання кращих границь. Але кінцеві результати виявилися майже однаковими при використанні обох прапорців.

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання №2: Обробка дисбалансу класів.

```
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

# 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, 'Training dataset')

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

# 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()
```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

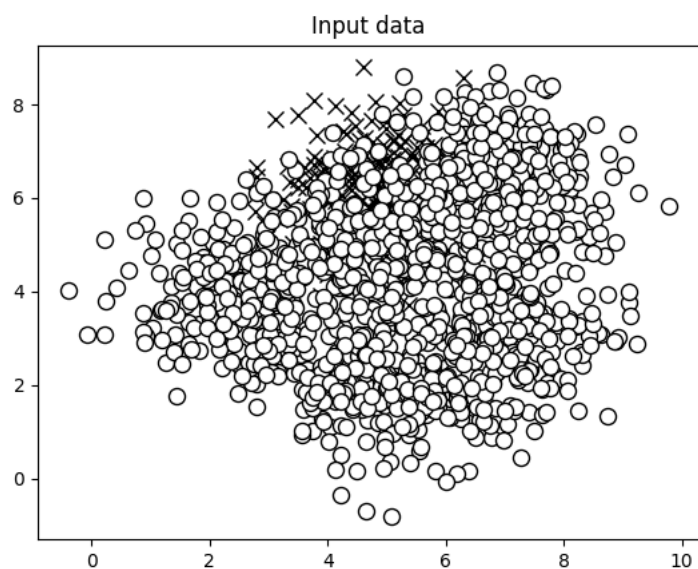


Рис. 12. Результат виконання програми

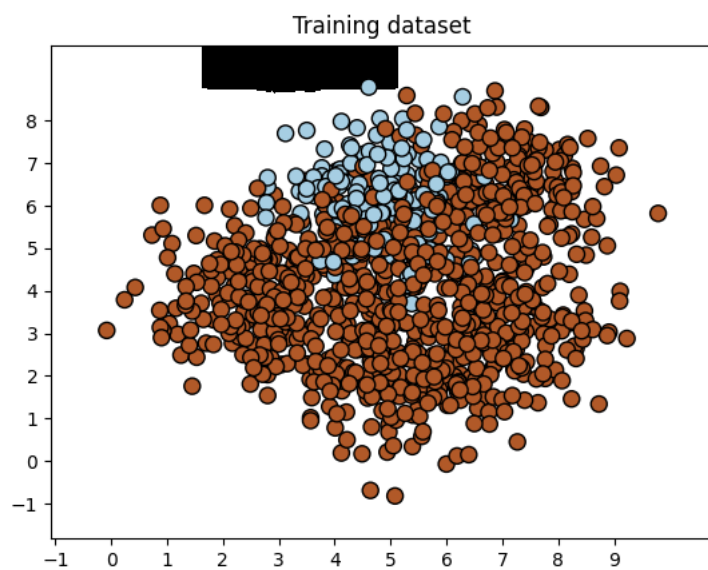


Рис. 13. Результат виконання програми

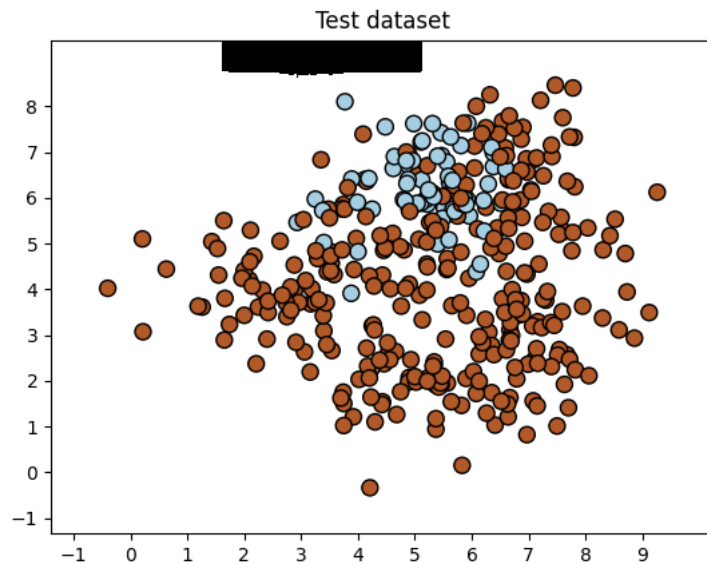


Рис. 14. Результат виконання програми

	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

#####

Рис. 15. Результат виконання програми

Завдання №3: Знаходження оптимальних навчальних параметрів за допомо-

гою сіткового пошуку.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
from utilities import visualize_classifier

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)

    df = pd.DataFrame(classifier.cv_results_)
    df_columns_to_print = [column for column in df.columns if 'param' in column or
                           'score' in column]
    print(df[df_columns_to_print])

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

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

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
##### Searching optimal parameters for precision_weighted
mean_score_time  std_score_time  ...  std_test_score  rank_test_score
0      0.002917      0.000044  ...      0.025132          1
1      0.002990      0.000030  ...      0.023032          5
2      0.003244      0.000025  ...      0.026560          4
3      0.003747      0.000086  ...      0.028334          8
4      0.003922      0.000135  ...      0.033429          9
5      0.001029      0.000012  ...      0.029698          2
6      0.001666      0.000012  ...      0.020401          7
7      0.002945      0.000013  ...      0.023032          5
8      0.011059      0.008402  ...      0.026867          3
```

[9 rows x 13 columns]

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

Рис. 16. Результат виконання програми

```
##### Searching optimal parameters for recall_weighted
      mean_score_time  std_score_time  ...  std_test_score  rank_test_score
0          0.002839      0.000064  ...      0.027075             1
1          0.003164      0.000099  ...      0.022468             5
2          0.003259      0.000059  ...      0.026749             3
3          0.003699      0.000015  ...      0.028497             8
4          0.003872      0.000039  ...      0.034744             9
5          0.001016      0.000007  ...      0.029407             1
6          0.001657      0.000006  ...      0.020096             7
7          0.002959      0.000046  ...      0.022468             5
8          0.006903      0.000036  ...      0.026749             3
```

[9 rows x 13 columns]

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

Рис. 17. Результат виконання програми

Завдання №4: Обчислення відносної важливості ознак.

```
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.model_selection import train_test_split
from sklearn.utils import shuffle

# Завантаження даних із цінами на нерухомість
housing_data = datasets.load_boston()

# 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 = housing_data.feature_names

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

# Сортювання 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[index_sorted])
plt.ylabel('Relative Importance')
plt.title('Feature importance using AdaBoost regressor')
plt.show()

```

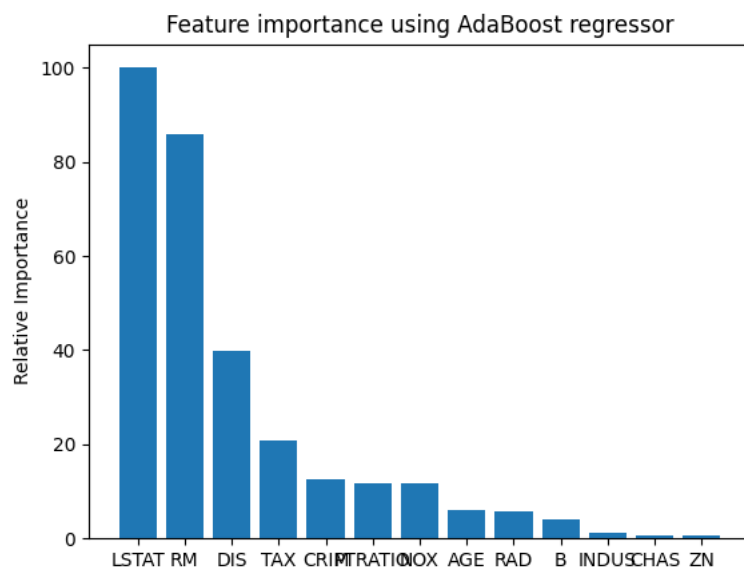


Рис. 18. Результат виконання програми

ADABOOST REGRESSOR

Mean squared error = 22.7

Explained variance score = 0.79

Рис. 19. Результат виконання програми

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання №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 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]]))
        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]))
```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Mean absolute error: 7.42
Predicted traffic: 26

Рис. 20. Результат виконання програми

Завдання №6: Створення навчального конвеєра (конвеєра машинного навчання).

```
from sklearn.datasets import _samples_generator
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.pipeline import Pipeline
from sklearn.ensemble import ExtraTreesClassifier

# Генерація даних
X, y = _samples_generator.make_classification(n_samples=150,
                                             n_features=25, n_classes=3,
                                             n_informative=6,
                                             n_redundant=0, random_state=7)

# Вибір k найважливіших ознак
k_best_selector = SelectKBest(f_regression, k=9)

# Ініціалізація класифікатора на основі гранично випадкового лісу
classifier = ExtraTreesClassifier(n_estimators=60, max_depth=4)

# Створення конвеєра
processor_pipeline = Pipeline([('selector', k_best_selector), ('erf',
                                                                classifier)])

# Встановлення параметрів
processor_pipeline.set_params(selector__k=7, erf__n_estimators=30)

# Навчання конвеєра
processor_pipeline.fit(X, y)

# Прогнозування результатів для вхідних даних
output = processor_pipeline.predict(X)
print("\nPredicted output:\n", output)

# Виведення оцінки
print("\nScore:", processor_pipeline.score(X, y))

# Виведення ознак, відібраних селектором конвеєра
status = processor_pipeline.named_steps['selector'].get_support()

# Вилучення та виведення індексів обраних ознак
selected = [i for i, x in enumerate(status) if x]
print("\nIndices of selected features:", ', '.join([str(x) for x in selected]))
```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```
/Library/Frameworks/Python.framework/Versions/Current/bin/python3 /Users/annasirach/
```

Predicted output:

```
[0 2 2 0 2 0 2 1 0 1 1 2 2 0 2 2 1 0 0 1 0 2 1 1 2 2 0 0 1 2 1 2 1 0 2 2 1
 1 2 2 2 0 1 2 2 1 2 2 1 0 1 2 2 2 2 0 2 2 0 2 2 0 1 0 2 2 1 1 1 2 0 1 0 2
 0 0 1 2 2 0 0 1 2 2 2 0 0 0 2 2 2 1 2 0 2 1 2 2 0 0 1 1 1 1 2 2 2 2 0 1 1
 0 2 1 1 0 1 1 1 1 0 0 0 1 2 1 0 0 2 1 2 0 0 1 0 1 1 0 1 1 1 1 2 0 0 1 2 0
 2 2]
```

Score: 0.9

Indices of selected features: 4, 7, 8, 12, 14, 17, 22

Рис. 21. Результат виконання програми

Висновок:

- В першому абзаці представлено спрогнозовані вихідні мітки за допомогою конвеєра.
- Значення Score відображає ефективність конвеєра.
- В останньому абзаці представлено індекси вибраних ознак.

Завдання №7: Пошук найближчих сусідів.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors

# Вхідні дані
X = np.array([[2.1, 1.3], [1.3, 3.2], [2.9, 2.5], [2.7, 5.4], [3.8, 0.9],
              [7.3, 2.1], [4.2, 6.5], [3.8, 3.7], [2.5, 4.1], [3.4, 1.9],
              [5.7, 3.5], [6.1, 4.3], [5.1, 2.2], [6.2, 1.1]])

# Кількість найближчих сусідів
k = 5

# Тестова точка даних
test_datapoint = [4.3, 2.7]

# Відображення вхідних даних на графіку
plt.figure()
plt.title('Input data')
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='black')

# Побудова моделі на основі методу k найближчих сусідів
knn_model = NearestNeighbors(n_neighbors=k, algorithm='ball_tree').fit(X)
distances, indices = knn_model.kneighbors([test_datapoint])

# Виведемо 'k' найближчих сусідів
print("\nK Nearest Neighbors:")
for rank, index in enumerate(indices[0][:k], start=1):
    print(str(rank) + " ==>", X[index])
```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Візуалізація найближчих сусідів разом із тестовою точкою даних
plt.figure()
plt.title('Nearest neighbors')
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='k')
plt.scatter(X[indices][0][:][:, 0], X[indices][0][:][:, 1],
            marker='o', s=250, color='k', facecolors='none')
plt.scatter(test_datapoint[0], test_datapoint[1],
            marker='x', s=75, color='k')
plt.show()
```

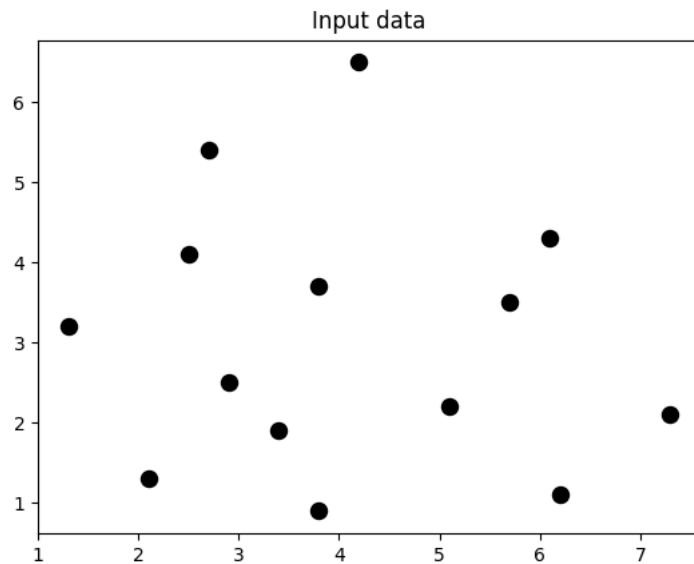


Рис. 22. Результат виконання програми

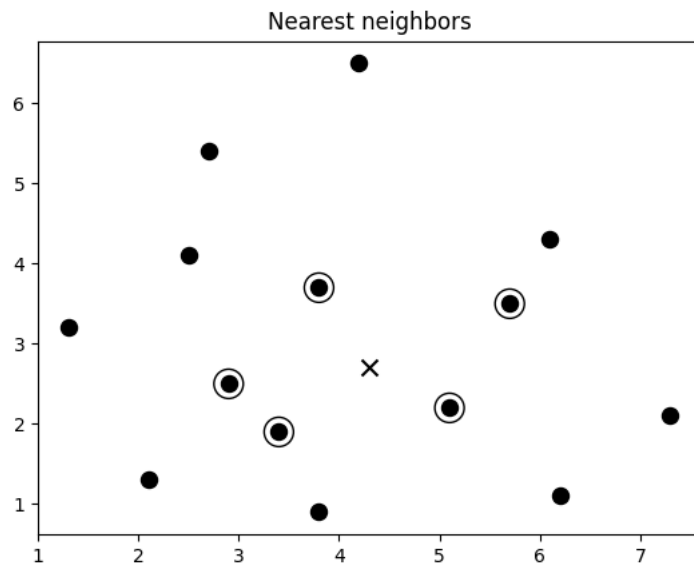


Рис. 23. Результат виконання програми

K Nearest Neighbors:

1 ==> [5.1 2.2]

2 ==> [3.8 3.7]

3 ==> [3.4 1.9]

4 ==> [2.9 2.5]

5 ==> [5.7 3.5]

Рис. 24. Результат виконання програми

Висновок:

- Перший скрін – це вхідні дані
- Другий скрін - це вхідні дані, тестова точка та її 5 найближчих сусідів (обведені)
- Третій скрін – це 5 найближчих сусідів

Завдання №8: Створити класифікатор методом k найближчих сусідів.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn import neighbors, datasets

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

# Відображення вхідних даних на графіку
plt.figure()
plt.title('Input data')
marker_shapes = 'v^os'
mapper = [marker_shapes[i] for i in y]
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=75, edgecolors='black', facecolors='none')

# Кількість найближчих сусідів
num_neighbors = 12

# Розмір кроку сітки візуалізації
step_size = 0.01

# Створення класифікатора на основі методу k найближчих сусідів
classifier = neighbors.KNeighborsClassifier(num_neighbors, weights='distance')

# Навчання моделі на основі методу k найближчих сусідів
classifier.fit(X, y)
```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Створення сітки для відображення меж на графіку
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_values, y_values = np.meshgrid(np.arange(x_min, x_max, step_size),
np.arange(y_min, y_max, step_size))

# Виконання класифікатора на всіх точках сітки
output = classifier.predict(np.c_[x_values.ravel(), y_values.ravel()])

# Візуалізація передбачуваного результату
output = output.reshape(x_values.shape)
plt.figure()
plt.pcolormesh(x_values, y_values, output, cmap=cm.Paired)

# Накладання навчальних точок на карту
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=50, edgecolors='black', facecolors='none')

plt.xlim(x_values.min(), x_values.max())
plt.ylim(y_values.min(), y_values.max())
plt.title('K Nearest Neighbors classifier model boundaries')

# Тестування вхідної точки даних
test_datapoint = [5.1, 3.6]
plt.figure()
plt.title('Test datapoint')
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=75, edgecolors='black', facecolors='none')

plt.scatter(test_datapoint[0], test_datapoint[1], marker='x',
            linewidth=6, s=200, facecolors='black')

# Вилучення K найближчих сусідів
_, indices = classifier.kneighbors([test_datapoint])
indices = indices.astype(np.int)[0]

# Відображення K найближчих сусідів на графіку
plt.figure()
plt.title('K Nearest Neighbors')

for i in indices:
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[y[i]],
                linewidth=3, s=100, facecolors='black')

plt.scatter(test_datapoint[0], test_datapoint[1], marker='x',
            linewidth=6, s=200, facecolors='black')

for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=75, edgecolors='black', facecolors='none')

print("Predicted output:", classifier.predict([test_datapoint])[0])

plt.show()

```

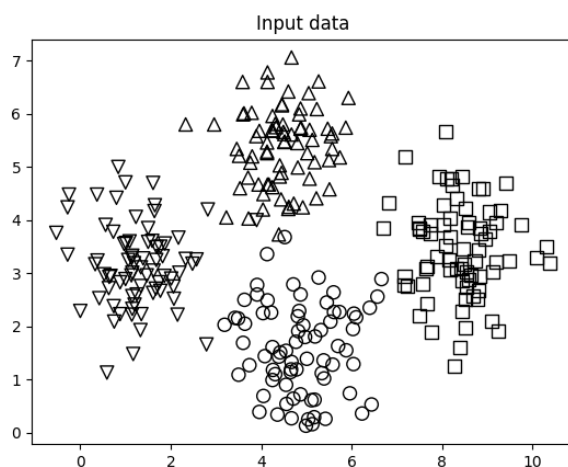


Рис. 25. Результат виконання програми

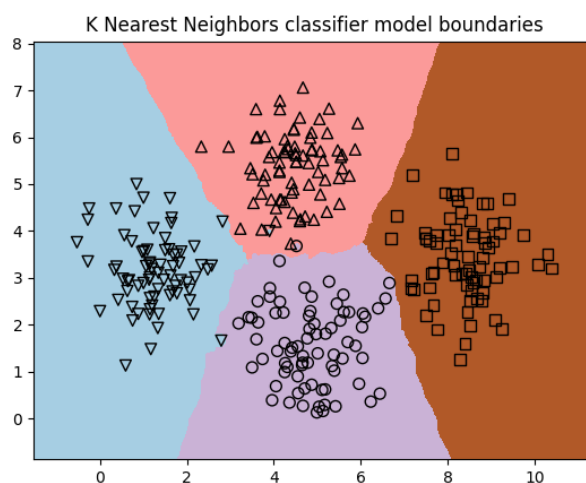


Рис. 26. Результат виконання програми

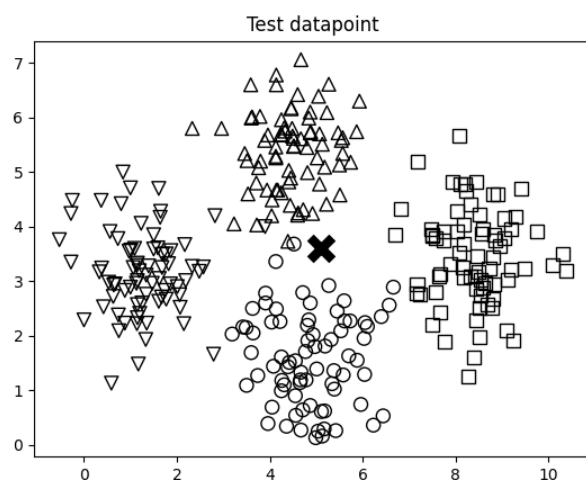


Рис. 27. Результат виконання програми

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

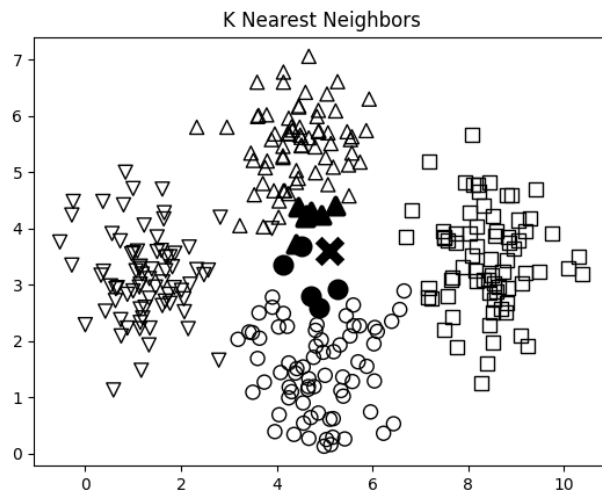


Рис. 28. Результат виконання програми

Predicted output: 1

Рис. 29. Результат виконання програми

Висновок:

- Перший скрін – це вхідні дані
- Другий - це межі класифікатора
- Третій - це тестова точка до вхідного набору даних
- Четвертий – це 12 найближчих сусідів
- Тестова точка – це 1 клас

Завдання №9: Обчислення оцінок подібності.

```
import argparse
import json
import numpy as np

def build_arg_parser():
    parser = argparse.ArgumentParser(description='Compute similarity score')
    parser.add_argument('--user1', dest='user1', required=True, help='First user')
    parser.add_argument('--user2', dest='user2', required=True,
                        help='Second user')
    parser.add_argument("--score-type", dest="score_type", required=True,
                        choices=['Euclidean', 'Pearson'], help='Similarity metric to be used')
```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return parser

# Обчислення оцінки евклідова відстані між користувачами user1 та user2
def euclidean_score(dataset, user1, user2):
    if user1 not in dataset:
        raise TypeError('Cannot find ' + user1 + ' in the dataset')

    if user2 not in dataset:
        raise TypeError('Cannot find ' + user2 + ' in the dataset')

    # Фільми, оцінені обома користувачами, user1 та user2
    common_movies = {}

    for item in dataset[user1]:
        if item in dataset[user2]:
            common_movies[item] = 1

    # За відсутності фільмів, оцінених обома користувачами, оцінка приймається
    # рівною 0
    if len(common_movies) == 0:
        return 0

    squared_diff = []

    for item in dataset[user1]:
        if item in dataset[user2]:
            squared_diff.append(np.square(dataset[user1][item] -
dataset[user2][item]))

    return 1 / (1 + np.sqrt(np.sum(squared_diff)))

# Обчислення кореляційної оцінки Пірсона між користувачем1 і користувачем2
def pearson_score(dataset, user1, user2):
    if user1 not in dataset:
        raise TypeError('Cannot find ' + user1 + ' in the dataset')

    if user2 not in dataset:
        raise TypeError('Cannot find ' + user2 + ' in the dataset')

    # Фільми, оцінені обома користувачами, user1 та user2
    common_movies = {}

    for item in dataset[user1]:
        if item in dataset[user2]:
            common_movies[item] = 1

    num_ratings = len(common_movies)

    # За відсутності фільмів, оцінених обома користувачами, оцінка приймається
    # рівною 0
    if num_ratings == 0:
        return 0

    # Обчислення суми рейтингових оцінок усіх фільмів, оцінених обома
    # користувачами
    user1_sum = np.sum([dataset[user1][item] for item in common_movies])
    user2_sum = np.sum([dataset[user2][item] for item in common_movies])

    # Обчислення суми квадратів рейтингових оцінок всіх фільмів, оцінених обома
    # користувачами
    user1_squared_sum = np.sum([np.square(dataset[user1][item]) for item in

```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				24
Змн.	Арк.	№ докум.	Підпис	Дата		


```

common_movies])
    user2_squared_sum = np.sum([np.square(dataset[user2][item]) for item in
common_movies])

    # Обчислення суми творів рейтингових оцінок всіх фільмів, оцінених обома
користувачами
    sum_of_products = np.sum([dataset[user1][item] * dataset[user2][item] for item
in common_movies])

    # Обчислення коефіцієнта кореляції Пірсона
    Sxy = sum_of_products - (user1_sum * user2_sum / num_ratings)
    Sxx = user1_squared_sum - np.square(user1_sum) / num_ratings
    Syy = user2_squared_sum - np.square(user2_sum) / num_ratings

    if Sxx * Syy == 0:
        return 0

    return Sxy / np.sqrt(Sxx * Syy)

if __name__ == '__main__':
    args = build_arg_parser().parse_args()
    user1 = args.user1
    user2 = args.user2
    score_type = args.score_type

    ratings_file = 'ratings.json'

    with open(ratings_file, 'r') as f:
        data = json.loads(f.read())

    if score_type == 'Euclidean':
        print("\nEuclidean score:")
        print(euclidean_score(data, user1, user2))
    else:
        print("\nPearson score:")
        print(pearson_score(data, user1, user2))

annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Bill Duffy" --score-type Euclidean

Euclidean score:
0.585786437626905
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Euclidean

Euclidean score:
0.30383243470068705
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Euclidean

Euclidean score:
0.2857142857142857
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Euclidean

```

Рис. 30. Результат виконання програми

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

```

annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Pearson

Pearson score:
0.6944217062199275
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Pearson

Pearson score:
0.6944217062199275
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Pearson

Pearson score:
0
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Pearson

Pearson score:
0.7587869106393281
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_9.py --user1 "David Smith" --user2 "Bill Duffy" --score-type Pearson

Pearson score:
0.9909924304103233

```

Рис. 31. Результат виконання програми

Висновок: оцінка подібності за Пірсоном демонструє кращі результати в порівнянні з евклідовою оцінкою подібності.

Завдання №10: Пошук користувачів зі схожими уподобаннями методом ко-лаборативної фільтрації.

```

import argparse
import json
import numpy as np

from LR_4_task_9 import pearson_score

def build_arg_parser():
    parser = argparse.ArgumentParser(description='Find users who are similar to the in-put user')
    parser.add_argument('--user', dest='user', required=True, help='Input user')
    return parser

# Знаходження користувачів у наборі даних, схожих на введеного користувача
def find_similar_users(dataset, user, num_users):
    if user not in dataset:
        raise TypeError('Cannot find ' + user + ' in the dataset')

    # Обчислення оцінки подібності за Пірсоном між
    # вказаним користувачем та всіма іншими
    # користувачами в наборі даних
    scores = np.array([x, pearson_score(dataset, user,
                                         x)] for x in dataset if x != user])

    # Сортвання оцінок за спаданням
    scores_sorted = np.argsort(scores[:, 1])[::-1]

    # Вилучення оцінок перших 'num_users' користувачів
    top_users = scores_sorted[:num_users]

    return scores[top_users]

```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

    ratings_file = 'ratings.json'

    with open(ratings_file, 'r') as f:
        data = json.loads(f.read())

    print('\nUsers similar to ' + user + ':\n')
    similar_users = find_similar_users(data, user, 3)
    print('User\t\t\tSimilarity score')
    print('-' * 41)
    for item in similar_users:
        print(item[0], '\t\t\t', round(float(item[1]), 2))

```

annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_10.py --user "David Smith"

Users similar to David Smith:

User	Similarity score
Chris Duncan	1.0
Bill Duffy	0.99
Adam Cohen	0.91

annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_10.py --user "Bill Duffy"

Users similar to Bill Duffy:

User	Similarity score
David Smith	0.99
Samuel Miller	0.88
Adam Cohen	0.86

annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_10.py --user "Clarissa Jackson"

Users similar to Clarissa Jackson:

User	Similarity score
Chris Duncan	1.0
Bill Duffy	0.83
Samuel Miller	0.73

Рис. 32. Результат виконання програми

Висновок: користувача Clarissa Jackson має однакові вподобання з користувачем Chris Duncan, а користувач Bill Duffy – майже однакові з David Smith.

Завдання №11: Створення рекомендаційної системи фільмів.

```

import argparse
import json
import numpy as np

from LR_4_task_9 import pearson_score
from LR_4_task_10 import find_similar_users

def build_arg_parser():

```

```

parser = argparse.ArgumentParser(description='Find the movie recommendations
for the given user')
parser.add_argument('--user', dest='user', required=True, help='Input user')
return parser

# Отримання рекомендації щодо фільмів для вказаного користувача
def get_recommendations(dataset, input_user):
    if input_user not in dataset:
        raise TypeError('Cannot find ' + input_user + ' in the dataset')

    overall_scores = {}
    similarity_scores = {}

    for user in [x for x in dataset if x != input_user]:
        similarity_score = pearson_score(dataset, input_user, user)

        if similarity_score <= 0:
            continue

        filtered_list = [x for x in dataset[user] if x not in \
            dataset[input_user] or dataset[input_user][x] == 0]

        for item in filtered_list:
            overall_scores.update({item: dataset[user][item] * similarity_score})
            similarity_scores.update({item: similarity_score})

    if len(overall_scores) == 0:
        return ['No recommendations possible']

    # Генерація рейтингів фільмів за допомогою їх нормалізації
    movie_scores = np.array([[score / similarity_scores[item], item]
        for item, score in overall_scores.items()])

    # Сортуювання за спаданням
    movie_scores = movie_scores[np.argsort(movie_scores[:, 0])[::-1]]

    # Вилучення рекомендацій фільмів
    movie_recommendations = [movie for _, movie in movie_scores]

    return movie_recommendations

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

    ratings_file = 'ratings.json'

    with open(ratings_file, 'r') as f:
        data = json.loads(f.read())

    print("\nMovie recommendations for " + user + ":")
    movies = get_recommendations(data, user)
    for i, movie in enumerate(movies):
        print(str(i + 1) + '. ' + movie)

```

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

```
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_11.py --user "Clarissa Jackson"
```

Movie recommendations for Clarissa Jackson:

1. No recommendations possible

```
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_11.py --user "Bill Duffy"
```

Movie recommendations for Bill Duffy:

1. Raging Bull

2. Roman Holiday

```
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_11.py --user "Bill Duffy"
```

Movie recommendations for Bill Duffy:

1. Raging Bull

2. Roman Holiday

```
annasirach@MacBook-Air-Anna Laba 4 % python3 -W ignore LR_4_task_11.py --user "Julie Hammel"
```

Movie recommendations for Julie Hammel:

1. The Apartment

2. Vertigo

3. Raging Bull

```
annasirach@MacBook-Air-Anna Laba 4 % █
```

Рис. 33. Результат виконання програми

Висновок:

- Для користувача Julie Hammel = 3 рекомендації
- Для користувача Bill Duffy = 2 рекомендації
- Для користувача Clarissa Jackson = 0 рекомендації

Висновок: Після виконання лаби навчився використовувати спеціалізовані бібліотеки та мову програмування Python дослідив методи ансамблів у машинному навчанні та створити рекомендаційні системи.

Посилання на GitHub: https://github.com/annasirach/AI_IPZ193_Sirach

		Сірач А.С.			ДУ «Житомирська політехніка».22.121.14.000 – Лр4	Арк.
		Філіпов В.О.				29
Змн.	Арк.	№ докум.	Підпис	Дата		