

Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ ТА СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

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

Хід роботи:

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

Лістинг програми:

```
import argparse
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
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__':
    args = build_arg_parser().parse_args()
    classifier_type = args.classifier_type

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

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

    plt.figure()
    plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='white',
                edgecolors='black', linewidth=1, marker='s')
    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='^')
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4									
Змн.	Арк.	№ докум.	Підпис	Дата										
Розроб.		Груницький Д.С.			Звіт з лабораторної роботи №4				Літ.		Арк.		Аркушів	
Перевір.		Голенко М.Ю.									1		32	
Реценз.									ФІКТ, гр.ІПЗ-20-3					
Н. Контр.														
Зав.каф.														

```

plt.title('Вхідні дані')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=5)
params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

if classifier_type == 'rf':
    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, 'Тестовий набір даних')

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('\nConfidense 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_classifier(classifier, test_datapoints, [0] * len(test_datapoints),
'Tестові точки даних')
plt.show()

```

Результат виконання програми із прапором **rf**:

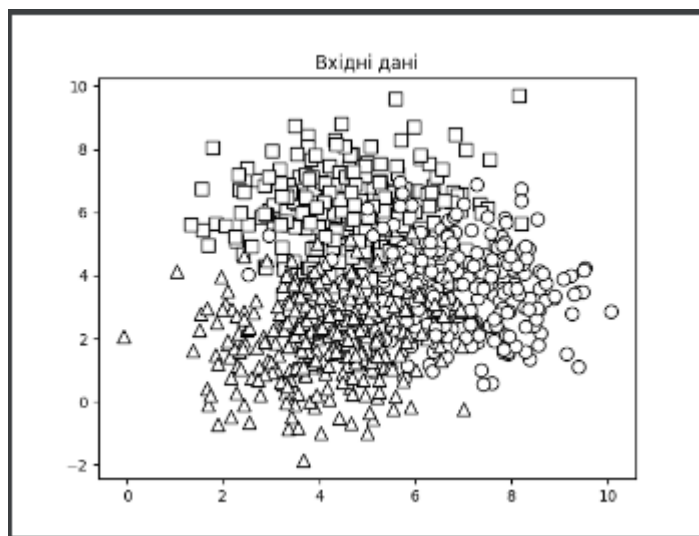


Рис. 2.1.1 – Результат виконання завдання (графік 1).

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

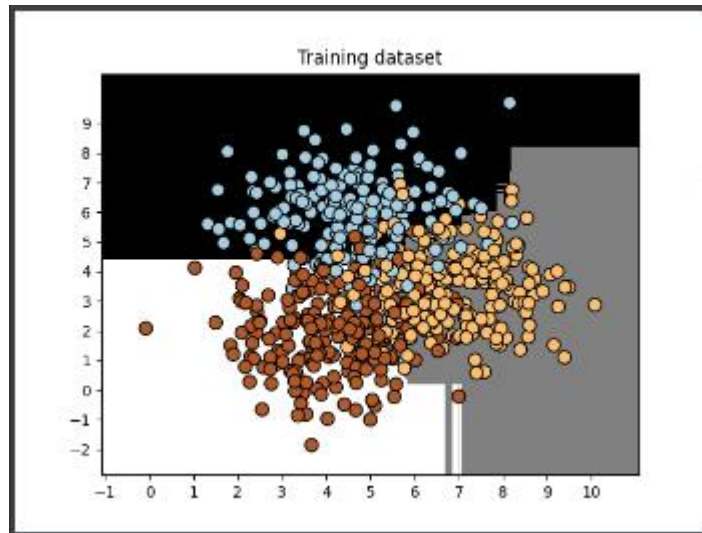


Рис. 2.1.2 – Результат виконання завдання (графік 2).

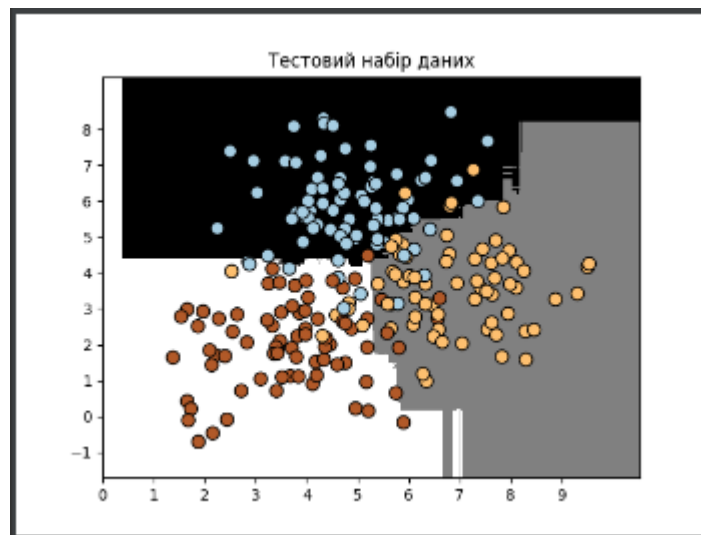


Рис. 2.1.3 – Результат виконання завдання (графік 3).

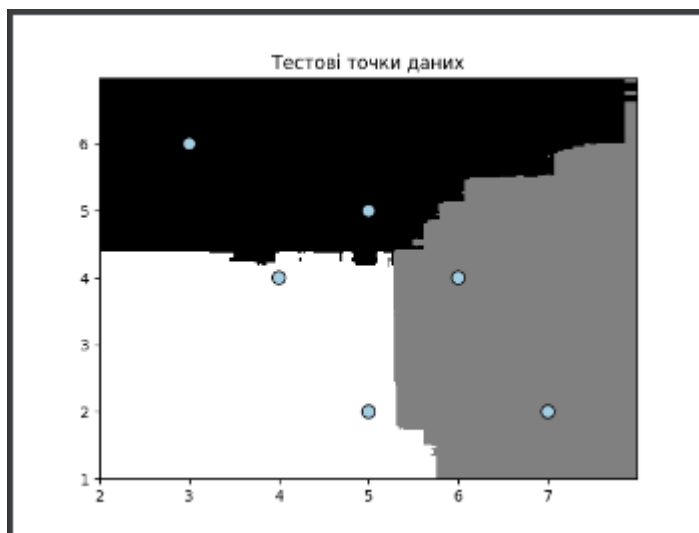


Рис. 2.1.4 – Результат виконання завдання (графік 4).

```
#####
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

Рис. 2.1.5 – Результат виконання завдання.

```
#####

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         225
weighted avg              0.87         225

#####
```

Рис. 2.1.6 – Результат виконання завдання.

```
Confidense 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. 2.1.7 – Результат виконання завдання.

Результат виконання програми із прапором **erf**:

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

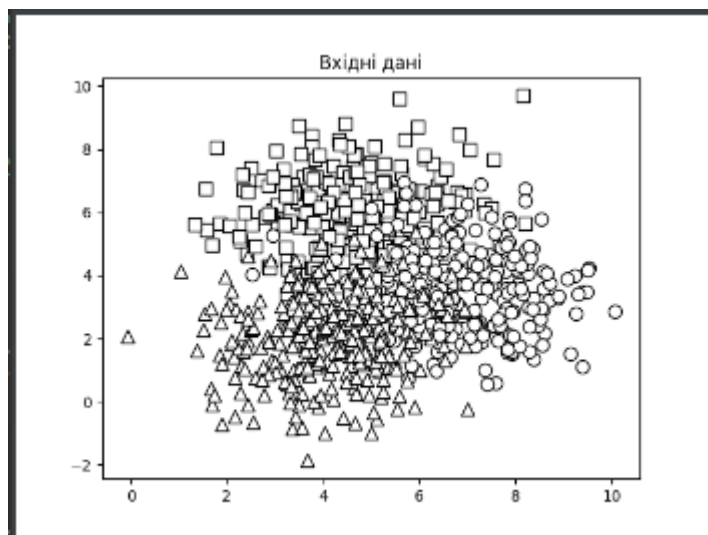


Рис. 2.1.8 – Результат виконання завдання (графік 5).

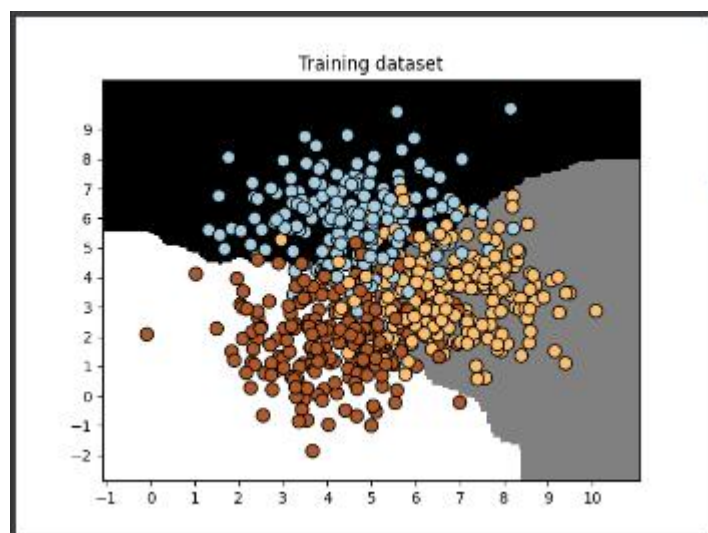


Рис. 2.1.9 – Результат виконання завдання (графік 6).

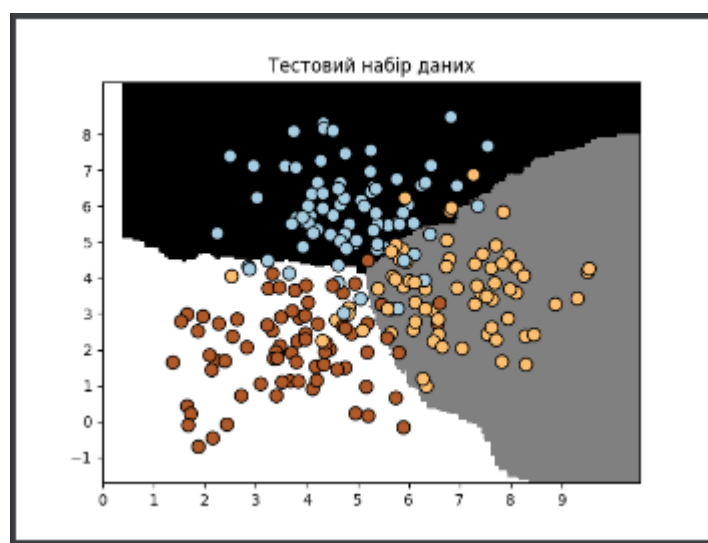


Рис. 2.1.10 – Результат виконання завдання (графік 7).

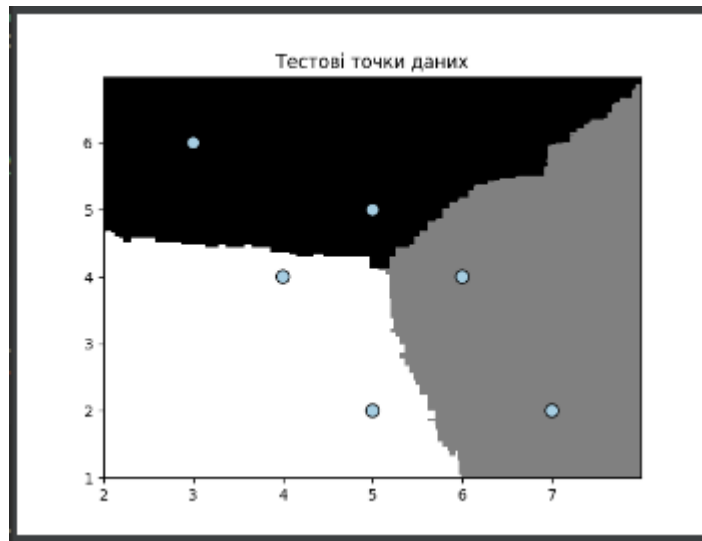


Рис. 2.1.11 – Результат виконання завдання (графік 8).

```
#####
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

Рис. 2.1.12 – Результат виконання завдання.

```
#####

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. 2.1.13 – Результат виконання завдання.

```
Confidense 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. 2.1.14 – Результат виконання завдання.

Висновок:

Висновок щодо класифікації подається для тренувального та тестового наборів даних.

Загальний результат класифікації показує, що класифікатори працюють досить добре на тренувальних і тестових даних. Метрики якості, такі як точність, відгук та $f1$ -показник, розраховані для кожного класу і для всього набору даних.

Для тестування впевненості моделі у передбаченнях, код також використовує класифікатор для передбачення класу для певних тестових точок даних і виводить впевненість у приналежності до класу.

Загальним висновок є те, що цей класифікатор є досить ефективним і може бути використаний для класифікації нових даних в один із трьох класів: Class-0, Class-1 або Class-2, з високою точністю та відгуком на тренувальних і тестових наборах даних.

Завдання 2.2. Обробка дисбалансу класів.

Лістинг програми:

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

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

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

plt.figure()
plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolor='black',
            linewidths=1, marker='x')

plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolor='white',
            edgecolors='black', linewidths=1, marker='o')

plt.title('Вхідні дані')

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

params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}
if len(sys.argv) > 1:
    if sys.argv[1] == 'balance':
```

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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 даних')

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

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

plt.show()

```

Результат виконання програми:

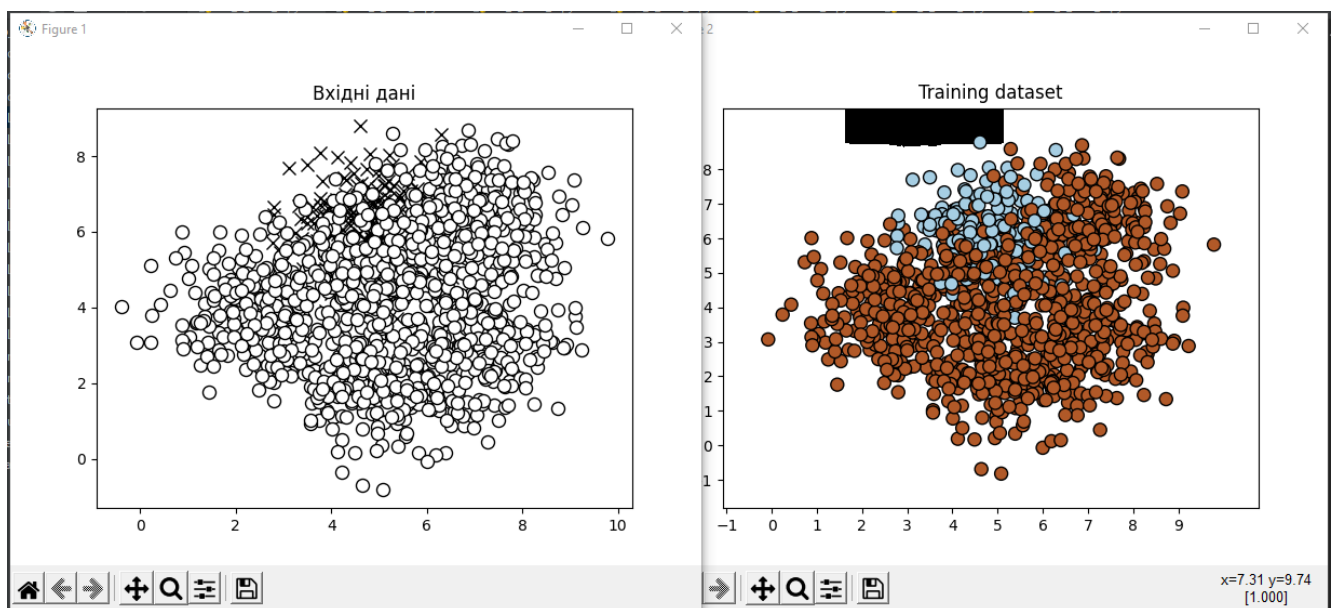


Рис. 2.2.1 – Результат виконання завдання (графік 1,2).

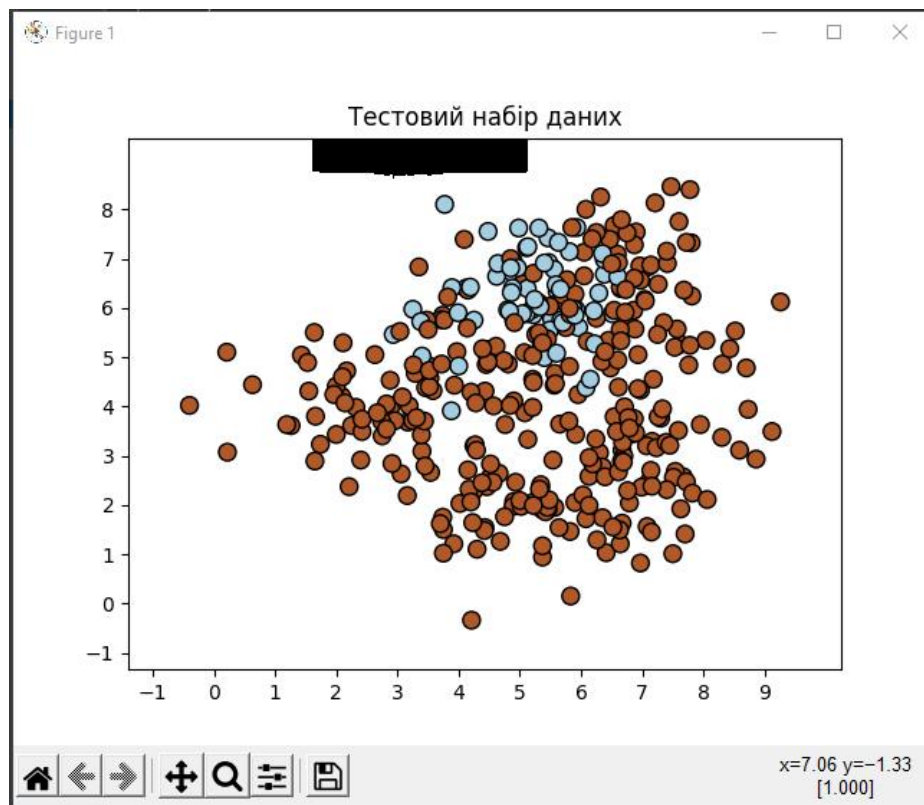


Рис. 2.2.2 – Результат виконання завдання (графік 3).

```
#####
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
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
macro avg       0.41      0.50      0.45      375
weighted avg     0.67      0.82      0.73      375

#####
```

Рис. 2.2.3 – Результат виконання завдання.

Результат виконання програми з врахуванням дисбалансу:

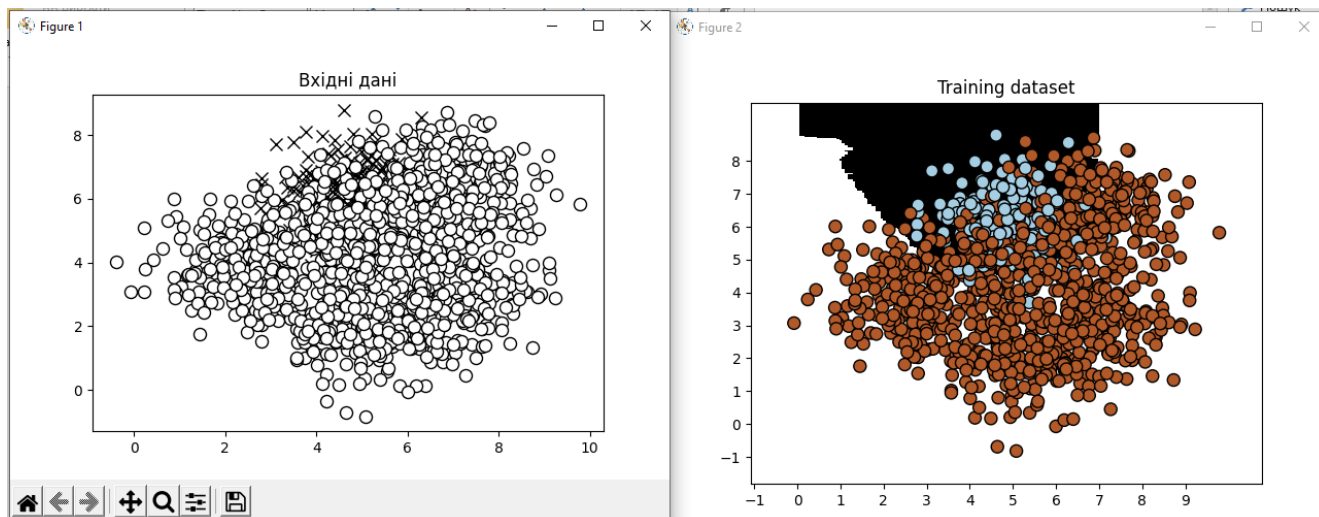


Рис. 2.2.4 – Результат виконання завдання (графік 4,5).

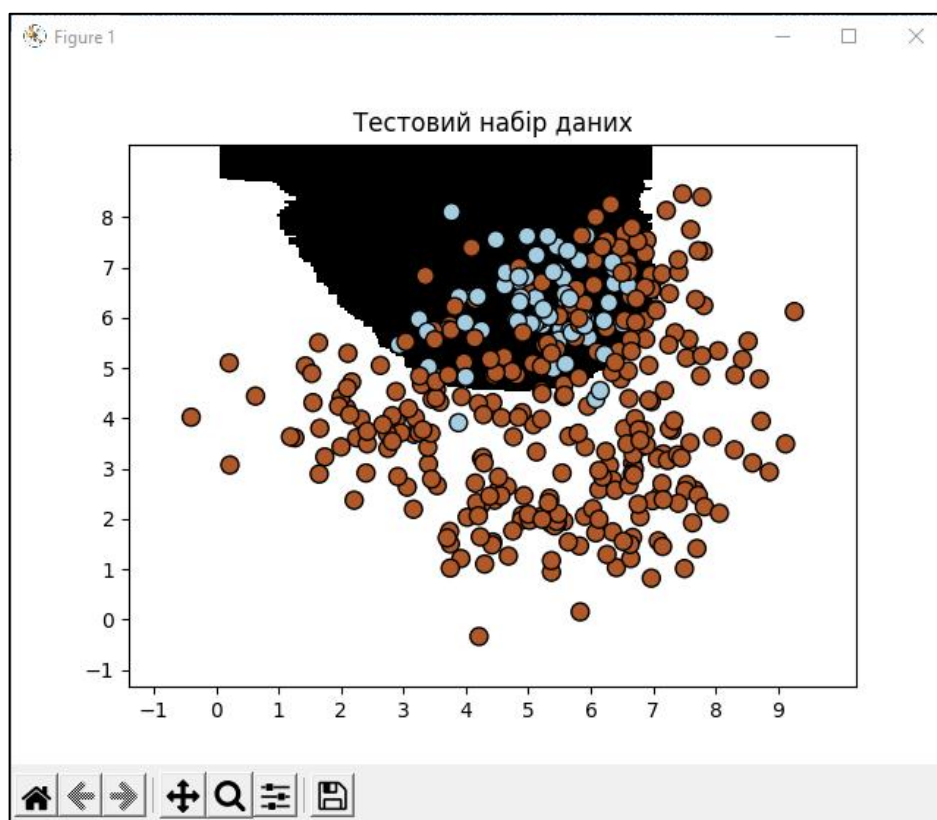


Рис. 2.2.5 – Результат виконання завдання (графік 6).

```
#####
Classifier performance on training dataset
      precision    recall  f1-score   support

   Class-0       0.44      0.93      0.60        181
   Class-1       0.98      0.77      0.86        944

 accuracy          0.80        1125
macro avg          0.71      0.85      0.73        1125
weighted avg       0.89      0.80      0.82        1125

#####
#####

Classifier performance on test dataset
      precision    recall  f1-score   support

   Class-0       0.45      0.94      0.61         69
   Class-1       0.98      0.74      0.84        306

 accuracy          0.78        375
macro avg          0.72      0.84      0.73        375
weighted avg       0.88      0.78      0.80        375

#####
```

Рис. 2.2.6 – Результат виконання завдання.

Висновок:

Модель може бути використана для класифікації цих даних, але важливо враховувати нерівноваженість класів та оптимізувати параметри моделі для поліпшення результатів.

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку.

Лістинг програми:

```
import numpy as np
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

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

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

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

```

random_state=5)

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

metrics = ['precision_weighted', 'recall_weighted']

for metric in metrics:
    print('\n#### Searching for optimal parameters for', metric)

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

    print('\nGrid scores for the parameter grid:')
    for i in range(0, len(classifier.cv_results_['params'])):
        print(classifier.cv_results_['params'][i], '-->',
              classifier.cv_results_['rank_test_score'][i])
    print('\nBest parameters:', classifier.best_params_)

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

```

Результат виконання програми:

```

#### Searching for optimal parameters for precision_weighted

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 1
{'max_depth': 4, 'n_estimators': 100} --> 5
{'max_depth': 7, 'n_estimators': 100} --> 4
{'max_depth': 12, 'n_estimators': 100} --> 8
{'max_depth': 16, 'n_estimators': 100} --> 9
{'max_depth': 4, 'n_estimators': 25} --> 2
{'max_depth': 4, 'n_estimators': 50} --> 7
{'max_depth': 4, 'n_estimators': 100} --> 5
{'max_depth': 4, 'n_estimators': 250} --> 3
|
Best parameters: {'max_depth': 2, 'n_estimators': 100}

```

Рис. 2.3.1 – Результат виконання завдання.

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		14


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

Grid scores for the parameter grid:
{'max_depth': 2, 'n_estimators': 100} --> 1
{'max_depth': 4, 'n_estimators': 100} --> 5
{'max_depth': 7, 'n_estimators': 100} --> 3
{'max_depth': 12, 'n_estimators': 100} --> 8
{'max_depth': 16, 'n_estimators': 100} --> 9
{'max_depth': 4, 'n_estimators': 25} --> 1
{'max_depth': 4, 'n_estimators': 50} --> 7
{'max_depth': 4, 'n_estimators': 100} --> 5
{'max_depth': 4, 'n_estimators': 250} --> 3

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

Рис. 2.3.2 – Результат виконання завдання.

```
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         0.86         0.86         225
 macro avg          0.86         0.86         0.86         225
weighted avg          0.86         0.86         0.86         225
```

Рис. 2.3.3 – Результат виконання завдання.

Висновок:

Результати пошуку параметрів показують, що найкращими параметрами для обидвох метрик були `{'max_depth': 2, 'n_estimators': 100}`. Після цього встановлені параметри були використані для прогнозування на тестовому наборі даних.

Звіт про продуктивність показує міру точності (ассирасу) приблизно 0.86 для всіх класів, а також інші метрики, такі як точність, чутливість та F1-середнє для кожного класу. Загальний висновок полягає в тому, що після налаштування параметрів Extra Trees Classifier, модель показує прийнятну точність на тестовому наборі даних.

Завдання 2.4. Обчислення відносної важливості ознак.

Лістинг програми:

```
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

housing_data = datasets.fetch_california_housing()

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=7)

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

y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)
print('\nADABOOST REGRESSOR')
print('Mean squared error =', round(mse, 2))
print('Explained variance error =', round(evs, 2))

feature_importances = regressor.feature_importances_
feature_names = housing_data.feature_names

feature_importances = 100.0 * (feature_importances / max(feature_importances))

index_sorted = np.flipud(np.argsort(feature_importances))
pos = np.arange(index_sorted.shape[0]) + 0.5

plt.figure()
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 важливості причин використання регресора AdaBoost')
plt.show()
```

Результат виконання програми:

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

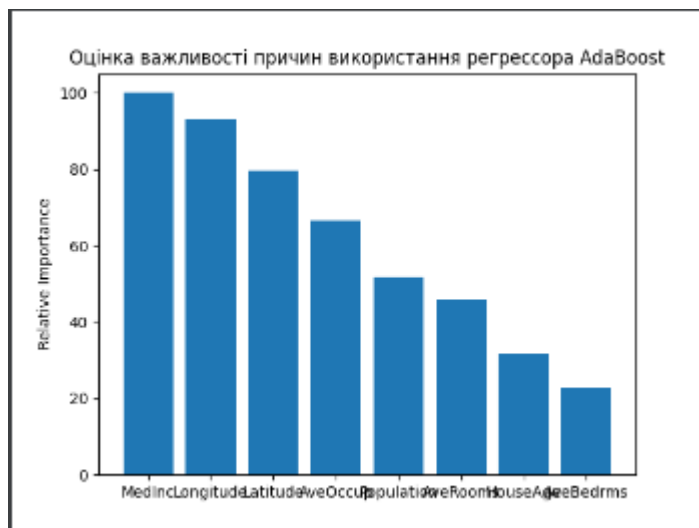


Рис. 2.4.1 – Результат виконання завдання (графік).

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

Рис. 2.4.2 – Результат виконання завдання.

Висновок:

Загальним висновком є те, що *AdaBoostRegressor* на базі рішучих дерев показав релативно низьку середньоквадратичну помилку і помірну пояснювальну варіацію, що свідчить про його відносну ефективність у прогнозуванні цін на нерухомість в Каліфорнії на основі наданих даних.

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

Лістинг програми:

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

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

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

data = np.array(data)

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)

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

params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}
regressor = ExtraTreesClassifier(**params)
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
print('Mean absolute error =', round(mean_absolute_error(y_test, y_pred), 2))

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:
        encoder = label_encoder[count]
        test_datapoint_encoded[i] = int(encoder.transform([test_datapoint[i]])[0])
        count = count + 1

test_datapoint_encoded = np.array(test_datapoint_encoded)
print('Predicted traffic:', int(regressor.predict([test_datapoint_encoded])[0]))

```

Результат виконання програми:

```

Mean absolute error = 13.35
Predicted traffic: 28

```

Рис. 2.5.1 – Результат виконання завдання.

Завдання 2.6. Створення навчального конвеєра (конвеєра машинного навчання).

Лістинг програми:

```

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

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

k_best_selector = SelectKBest(f_regression, k=9)

classifier = ExtraTreesClassifier(n_estimators=60, max_depth=4)

```

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

```

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

processor_pipeline.set_params(selector__k=7, erf__n_estimators=30)

processor_pipeline.fit(X, y)

output = processor_pipeline.predict(X)
print("\nPredicted output:\n", output)

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

status = processor_pipeline.named_steps['selector'].get_support()

selected = [i for i, x in enumerate(status) if x]
print("\nIndices of selected features:", ', '.join([str(x) for x in selected]))

```

Результат виконання програми:

```

Predicted output:
[1 2 2 0 2 0 2 1 0 1 1 2 2 0 2 2 1 0 1 1 0 2 0 1 2 2 0 0 1 1 1 0 1 0 2 2 1
 1 2 2 2 0 1 0 2 1 1 2 1 0 1 2 2 2 2 0 2 2 2 0 1 0 2 2 1 1 1 2 0 1 0 2
 0 0 1 2 2 0 0 2 2 2 0 0 0 0 2 2 2 1 2 0 2 0 2 2 0 0 1 1 1 1 2 2 1 2 0 1 1
 0 2 1 0 0 1 1 1 1 0 0 0 1 2 0 0 0 2 1 2 0 0 0 0 1 1 0 1 1 1 1 2 0 0 1 2 0
 2 2]

Score: 0.88

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

```

Рис. 2.6.1 – Результат виконання завдання.

1. У першому списку "Predicted output" містяться передбачені класи (мітки) для кожного прикладу з вхідних даних X . Кожне значення в цьому списку вказує на приналежність вхідного прикладу до одного з трьох класів.
2. Значення "Score" дорівнює 0.88, що вказує на точність моделі при передбаченні класів на вхідних даних X . Це означає, що модель правильно передбачила класи для 88% прикладів в наборі даних.
3. У останньому рядку "Indices of selected features" містяться індекси вибраних ознак, які були обрані в результаті використання методу відбору ознак

"SelectKBest". У цьому випадку було обрано 7 ознак з набору даних. Ці ознаки вважаються найбільш інформативними для моделі при класифікації.

Завдання 2.7. Пошук найближчих сусідів.

Лістинг програми:

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

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]])

k = 5

test_datapoint = [4.3, 2.7]

plt.figure()
plt.title('Вхідні дані')
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='black')

knn_model = NearestNeighbors(n_neighbors=k, algorithm='ball_tree').fit(X)
distances, indices = knn_model.kneighbors([test_datapoint])

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

plt.figure()
plt.title('Найближчі сусіди')
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='k')
plt.scatter(X[indices[0][0][:k][:, 0], X[indices[0][0][:k][:, 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()
```

Результат виконання програми:

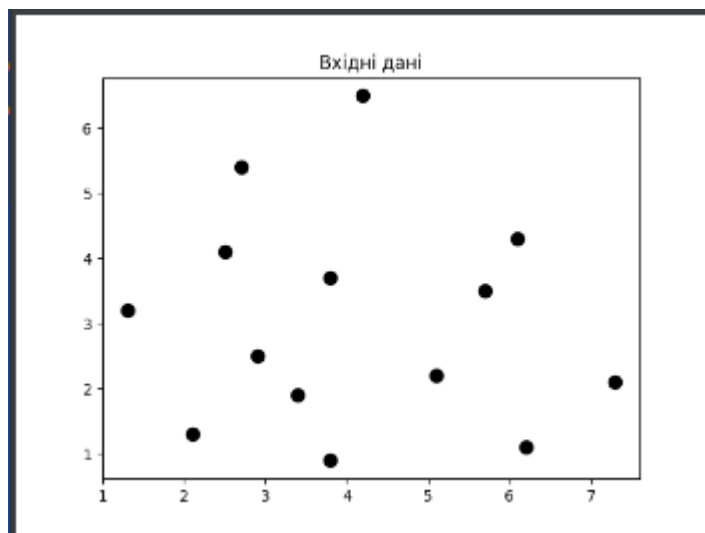


Рис. 2.7.1 – Результат виконання завдання (графік 1).

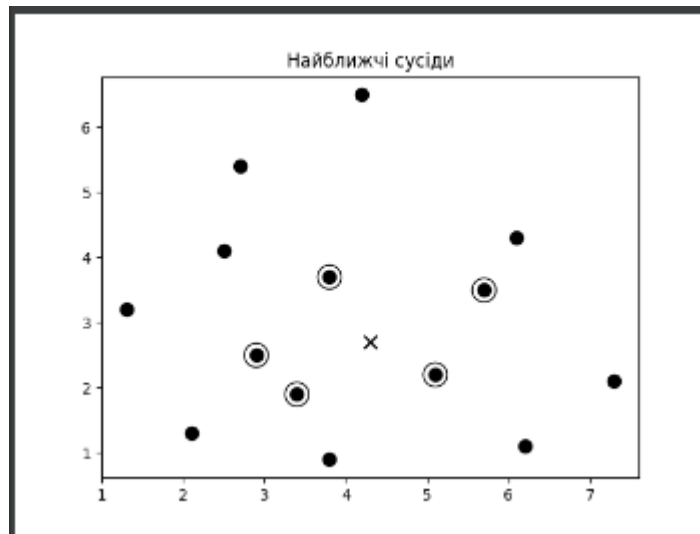


Рис. 2.7.2 – Результат виконання завдання (графік 2).

```
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. 2.7.3 – Результат виконання завдання.

Висновок:

1. На першому графіку відображено вхідні дані у вигляді точок.
2. На другому графіку відображено найближчих 5 сусідів тестової точки даних.
3. В терміналі відображено список з 5 найближчих сусідів.

Завдання 2.8. Створити класифікатор методом k найближчих сусідів.

Лістинг програми:

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

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

```

plt.figure()
plt.title('Вхідні дані')
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')

num_neighbors = 12

step_size = 0.01

classifier = neighbors.KNeighborsClassifier(num_neighbors, weights='distance')

classifier.fit(X, y)

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))

output = classifier.predict(np.c_[x_values.ravel(), y_values.ravel()])

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

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('Кордони моделі класифікатора на основі K найближчих сусідів')

test_datapoint = [5.1, 3.6]
plt.figure()
plt.title('Тестова точка даних')
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')

_, indices = classifier.kneighbors([test_datapoint])
indices = indices.astype(int)[0]

plt.figure()
plt.title('K найближчих сусідів')

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])

```

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

```
plt.show()
```

Результат виконання програми:

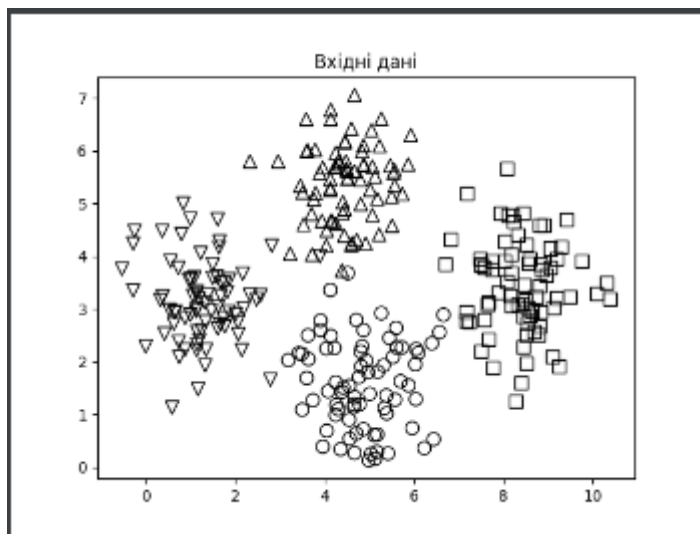


Рис. 2.8.1 – Результат виконання завдання (графік 1).

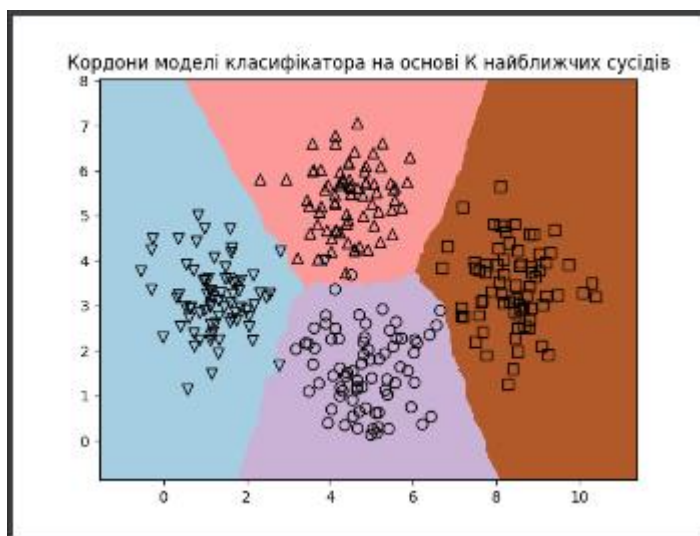


Рис. 2.8.2 – Результат виконання завдання (графік 2).

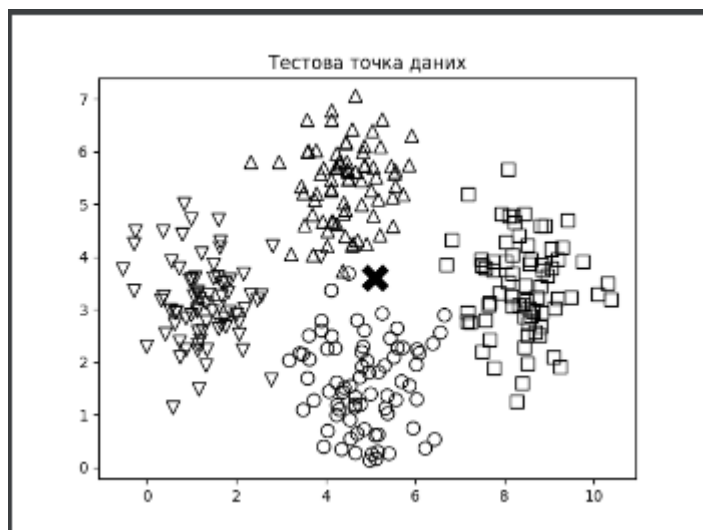


Рис. 2.8.3 – Результат виконання завдання (графік 3).

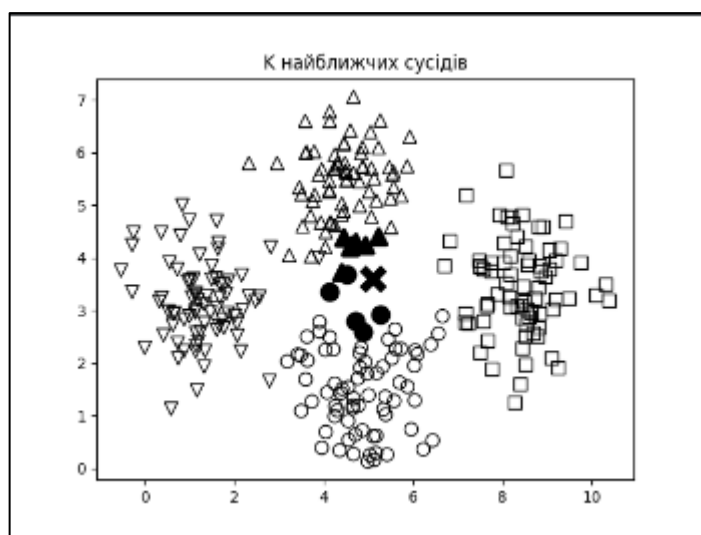


Рис. 2.8.4 – Результат виконання завдання (графік 4).

Predicted output: 1

Рис. 2.8.5 – Результат виконання завдання.

Висновок:

1. На першому графіку відображено вхідні дані.
2. На другому графіку, використовуючи метод k найближчих сусідів, модель визначає передбачувані межі між класами.
3. На третьому графіку відображено тестову точку та всі навчальні точки.
4. На четвертому графіку відображено K найближчих сусідів тестової точки.

5. Тестова точка відноситься до класу 1.

Завдання 2.9. Обчислення оцінок подібності.

Лістинг програми:

```
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')
    return parser

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')

    common_movies = {}

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

    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)))

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')

    common_movies = {}

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

    num_ratings = len(common_movies)

    if num_ratings == 0:
        return 0
```

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

```

user1_sum = np.sum([dataset[user1][item] for item in common_movies])
user2_sum = np.sum([dataset[user2][item] for item in common_movies])

user1_squared_sum = np.sum([np.square(dataset[user1][item]) for item in
common_movies])
user2_squared_sum = np.sum([np.square(dataset[user2][item]) for item in
common_movies])

sum_of_products = np.sum([dataset[user1][item] * dataset[user2][item] for item
in common_movies])

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

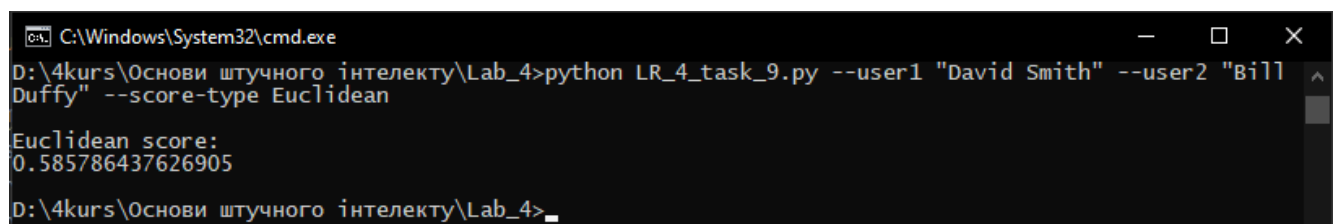
    ratings_file = 'ratings.json'

    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))

```

Результат виконання програми:



```

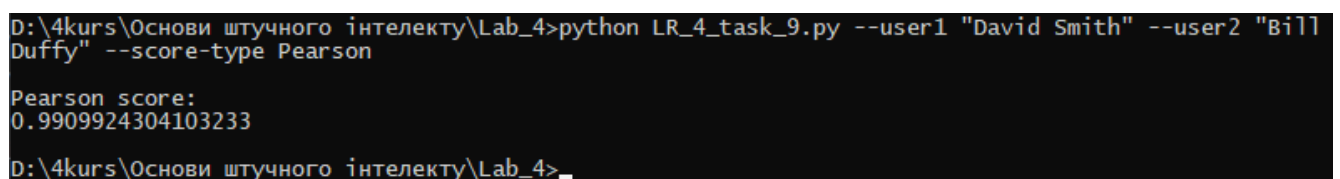
C:\Windows\System32\cmd.exe
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Bill
Duffy" --score-type Euclidean

Euclidean score:
0.585786437626905

D:\4kurs\Основи штучного інтелекту\Lab_4>_

```

Рис. 2.9.1 – Результат виконання завдання.



```

D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Bill
Duffy" --score-type Pearson

Pearson score:
0.9909924304103233

D:\4kurs\Основи штучного інтелекту\Lab_4>_

```

Рис. 2.9.2 – Результат виконання завдання.

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

```
C:\Windows\System32\cmd.exe

D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Euclidean

Euclidean score:
0.30383243470068705

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.3 – Результат виконання завдання.

```
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Pearson

Pearson score:
0.7587869106393281

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.4 – Результат виконання завдання.

```
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Euclidean

Euclidean score:
0.2857142857142857

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.5 – Результат виконання завдання.

```
C:\Windows\System32\cmd.exe

D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Pearson

Pearson score:
0

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.6 – Результат виконання завдання.

```
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Euclidean

Euclidean score:
0.28989794855663564

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.7 – Результат виконання завдання.

```
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Pearson

Pearson score:
0.6944217062199275

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.8 – Результат виконання завдання.

```
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Adam Cohen" --score-type Euclidean

Euclidean score:
0.38742588672279304

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.9 – Результат виконання завдання.

```
D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_9.py --user1 "David Smith" --user2 "Adam Cohen" --score-type Pearson

Pearson score:
0.9081082718950217

D:\4kurs\Основи штучного інтелекту\Lab_4>
```

Рис. 2.9.10 – Результат виконання завдання.

Висновок:

Можливість порівнювати користувачів на основі їхніх рейтингів для фільмів і визначити, наскільки вони схожі за допомогою обраної метрики.

Завдання 2.10. Пошук користувачів зі схожими уподобаннями методом колаборативної фільтрації.

Лістинг програми:

```
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 input user')
    parser.add_argument('--user', dest='user', required=True, help='Input user')
    return parser

def find_similar_users(dataset, user, num_users):
    if user not in dataset:
        raise TypeError('Cannot find ' + user + ' in the dataset')

    scores = np.array([x, pearson_score(dataset, user, x)] for x in dataset if x != user)

    scores_sorted = np.argsort(scores[:, 1])[:, :-1]

    top_users = scores_sorted[:num_users]
```

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return scores[top_users]

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))

```

Результат виконання програми:

```

D:\4kurs\Основи штучного інтелекту\Lab_4>python 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
D:\4kurs\Основи штучного інтелекту\Lab_4>

```

Рис. 2.10.1 – Результат виконання завдання (Bill Duffy).

```

D:\4kurs\Основи штучного інтелекту\Lab_4>python 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
D:\4kurs\Основи штучного інтелекту\Lab_4>_

```

Рис. 2.10.2 – Результат виконання завдання (Clarissa Jackson).

Висновок:

Результат виконання коду показує схожих користувачів для введеного користувача. Наприклад, у першому наборі виводу користувач "David Smith" має найвищий коефіцієнт схожості 0.99, що означає високу схожість з введеним користувачем. У другому наборі виводу користувач "Chris Duncan" має найвищий коефіцієнт схожості 1.0, що свідчить про ідентичність з введеним користувачем.

Отже, цей код допомагає знайти користувачів, схожих на введеного користувача, на основі їх рейтингів, та виводить їх на екран разом із значеннями схожості.

Завдання 2.11. Створення рекомендаційної системи фільмів.

Лістинг програми:

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

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']

    movie_scores = np.array([[score / similarity_scores[item], item]
                             for item, score in overall_scores.items()])

    movie_scores = movie_scores[np.argsort(movie_scores[:, 0])[:, :-1]]

    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'
```

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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)

```

Результат виконання програми:

```

C:\Windows\System32\cmd.exe

D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_11.py --user "Chris Duncan"

Movie recommendations for Chris Duncan:
1. Vertigo
2. Scarface
3. Goodfellas
4. Roman Holiday

D:\4kurs\Основи штучного інтелекту\Lab_4>

```

Рис. 2.11.1 – Результат виконання завдання (Chris Duncan).

```

C:\Windows\System32\cmd.exe

D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_11.py --user "Julie Hammel"

Movie recommendations for Julie Hammel:
1. The Apartment
2. Vertigo
3. Raging Bull

D:\4kurs\Основи штучного інтелекту\Lab_4>

```

Рис. 2.11.2 – Результат виконання завдання (Julie Hammel).

```

C:\Windows\System32\cmd.exe

D:\4kurs\Основи штучного інтелекту\Lab_4>python LR_4_task_11.py --user "Clarissa Jackson"

Movie recommendations for Clarissa Jackson:
1. No recommendations possible

D:\4kurs\Основи штучного інтелекту\Lab_4>

```

Рис. 2.11.3 – Результат виконання завдання (Clarissa Jackson).

Висновок:

Рекомендації фільмів засновані на подібності між цим користувачем та іншими користувачами. Таким чином, код виконує функцію рекомендації фільмів для введеного користувача на основі аналізу рейтингів користувачів та подібності користувачів.

Посилання на репозиторій: https://github.com/GrunytskyDmytro/Lab4_AI.git

Висновок по лабораторній роботі: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив методи ансамблів у машинному навчанні та створив рекомендаційні системи.

		Груницький Д.С.			ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.6.000 – Лр.4	Арк.
		Голенко М.Ю.				32
Змн.	Арк.	№ докум.	Підпис	Дата		