

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.

Хід роботи:

Завдання 1. Створення класифікаторів на основі випадкових та гранично випадкових лісів

Лістинг коду файлу Task_1.py:

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

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]
    print(X)

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

    plt.figure()
```

					ДУ «Житомирська політехніка».19.121.22.000 – Лр4			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Чижмотря М.О.			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Пулеко І.В.						Аркушів
Керівник								1
Н. контр.								21
Зав. каф.							ФІКТ Гр. ІПЗ-19-1[2]	

```

plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='red',
edgecolors='black', linewidth=1, marker='s')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='green',
edgecolors='black', linewidth=1, marker='o')
plt.scatter(class_2[:, 0], class_2[:, 1], s=75, facecolors='blue',
edgecolors='black', linewidth=1, marker='^')

plt.title('Input data')
plt.show()

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=5)
params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

if 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')

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

print("#" * 40)
print("\nClassifier performance on test dataset\n")
Y_test_pred = classifier.predict(X_test)
print(classification_report(Y_test, Y_test_pred, target_names=class_names))
print("#" * 40 + "\n")

```

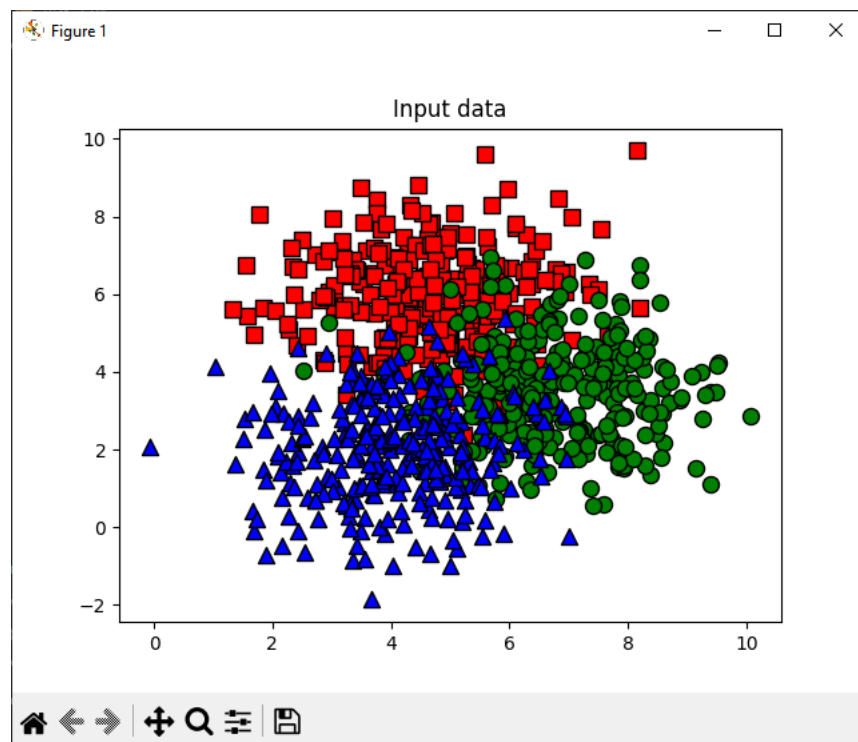


Рис.4.1 – Зображення розподілення даних

		Чижмоторя М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

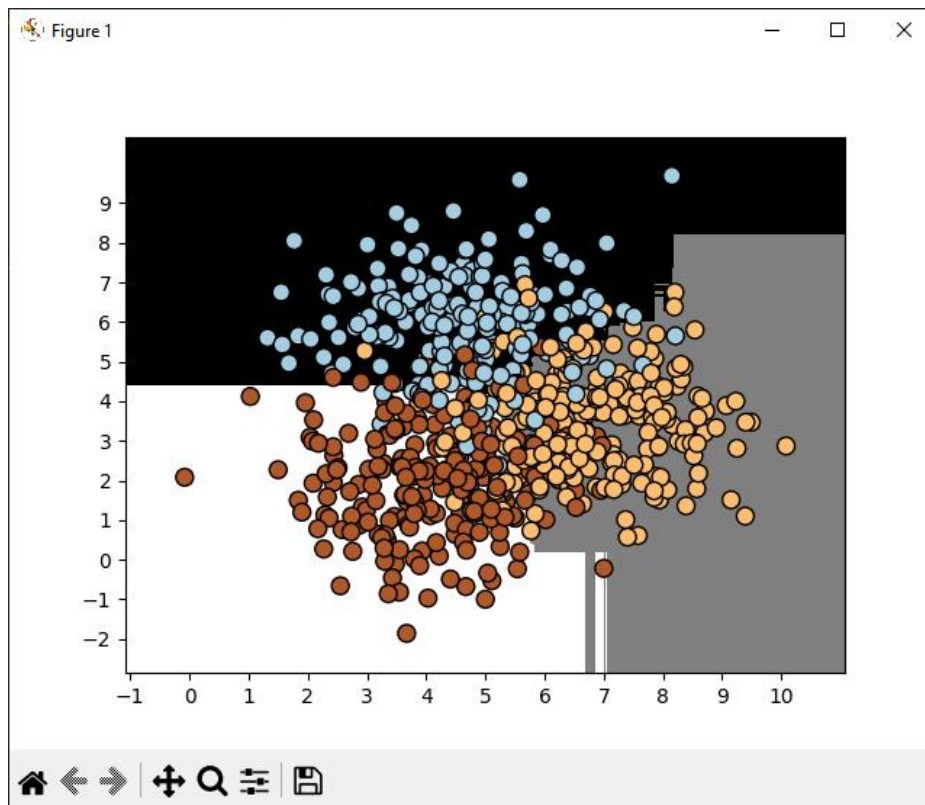


Рис.4.2 – Класифікація методом випадкових дерев

```
#####

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.4.3 – Характеристики роботи методу випадкових дерев

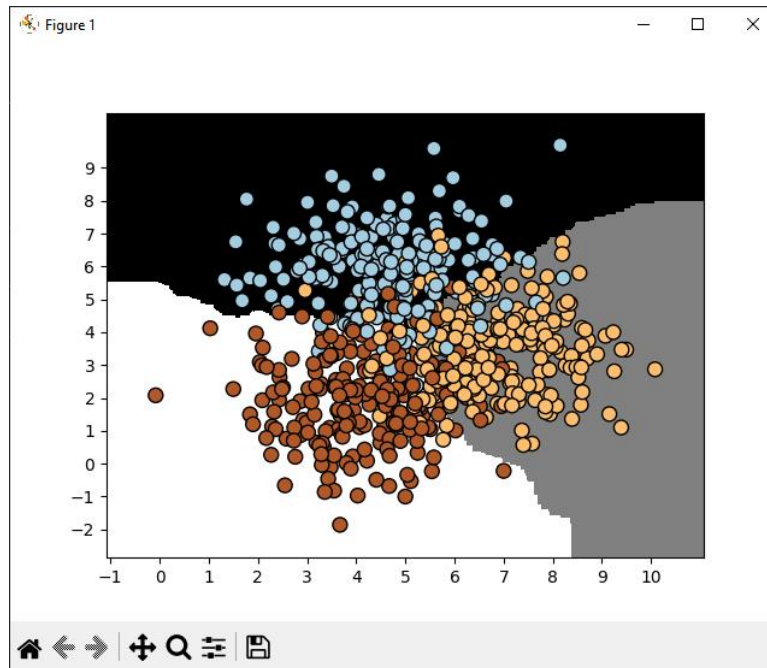


Рис.4.5 – Класифікація методом гранично випадкових дерев

```
#####
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.4.5 – Характеристики роботи методу гранично випадкових дерев

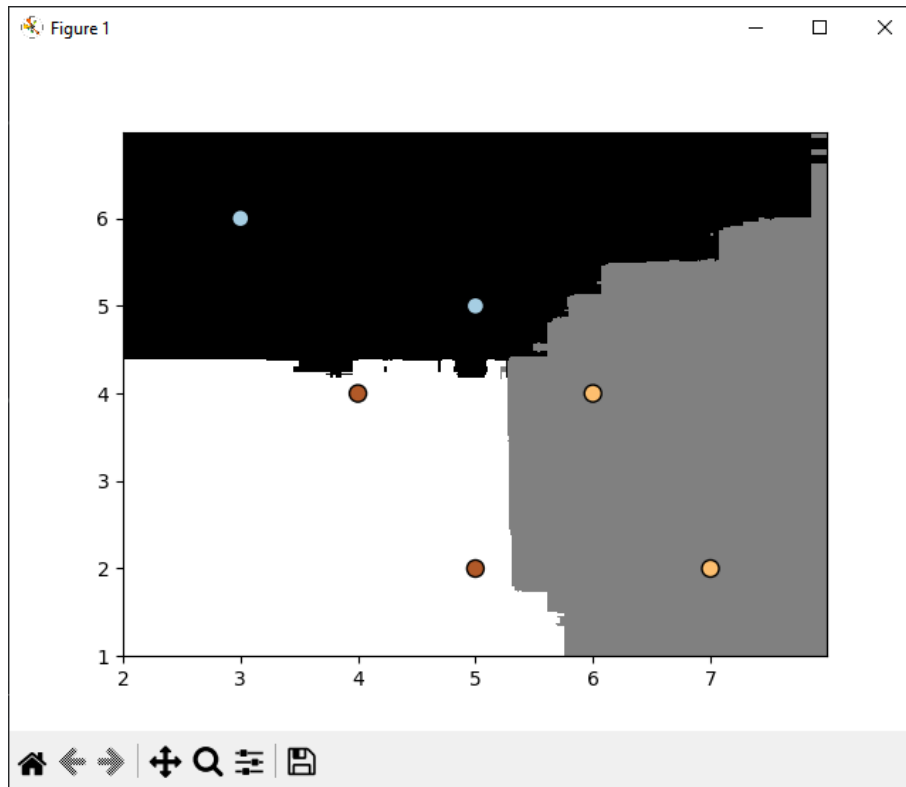


Рис.4.6 – Візуалізація можливих класів точок (rf)

```
Confidence measure:
Datapoint: [5 5]
Predicted class: Class-0
Probabilities: [0.81427532 0.08639273 0.09933195]
Datapoint: [3 6]
Predicted class: Class-0
Probabilities: [0.93574458 0.02465345 0.03960197]
Datapoint: [6 4]
Predicted class: Class-1
Probabilities: [0.12232404 0.7451078 0.13256816]
Datapoint: [7 2]
Predicted class: Class-1
Probabilities: [0.05415465 0.70660226 0.23924309]
Datapoint: [4 4]
Predicted class: Class-2
Probabilities: [0.20594744 0.15523491 0.63881765]
Datapoint: [5 2]
Predicted class: Class-2
Probabilities: [0.05403583 0.0931115 0.85285267]
```

Рис.4.7 – Дані про можливі класи (rf)

		Чижмотря М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

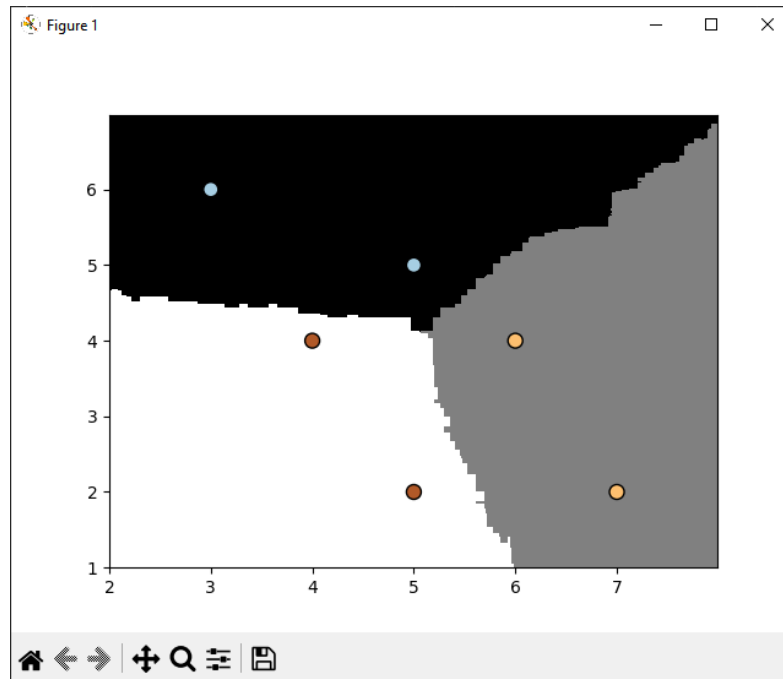


Рис.4.8 – Візуалізація можливих класів точок (erf)

```
Confidence measure:
Datapoint: [5 5]
Predicted class: Class-0
Probabilities: [0.48904419 0.28020114 0.23075467]
Datapoint: [3 6]
Predicted class: Class-0
Probabilities: [0.66707383 0.12424406 0.20868211]
Datapoint: [6 4]
Predicted class: Class-1
Probabilities: [0.25788769 0.49535144 0.24676087]
Datapoint: [7 2]
Predicted class: Class-1
Probabilities: [0.10794013 0.6246677 0.26739217]
Datapoint: [4 4]
Predicted class: Class-2
Probabilities: [0.33383778 0.21495182 0.45121039]
Datapoint: [5 2]
Predicted class: Class-2
Probabilities: [0.18671115 0.28760896 0.52567989]
```

Рис.4.9 – Дані про можливі класи (erf)

Використання випадкових дерев та граничних випадкових дерев дозволяє ефективно класифікувати дані, з двох методів останній має кращий результат.

		Чижмоторя М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2. Обробка дисбалансу класів

Лістинг коду файлу Task_2.py:

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

if __name__ == '__main__':
    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, 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')

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=5)
    params = {'n_estimators': 100, 'max_depth': 4, 'random_state': 0}

    if len(sys.argv) > 1:
        if sys.argv[1] == 'balance':
            params['class_weight'] = 'balanced'
        else:
            raise TypeError("Invalid input argument; should be 'balance' or
nothing")

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

    Y_test_pred = classifier.predict(X_test)
    class_names = ['Class-0', 'Class-1']
    print("\n" + "#" * 40)
    print("Classifier performance on training dataset")
    print(classification_report(Y_train, Y_train_pred, target_names=class_names))
    print("#" * 40)
    print("Classifier performance on test dataset")
    print(classification_report(Y_test, Y_test_pred, target_names=class_names))
    print("#" * 40 + "\n")
    plt.show()
```

		Чижмоторя М.О.			ДУ «Житомирська політехніка».19.121.2.000 - Лр4	Арк.
		Пулеко І.В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

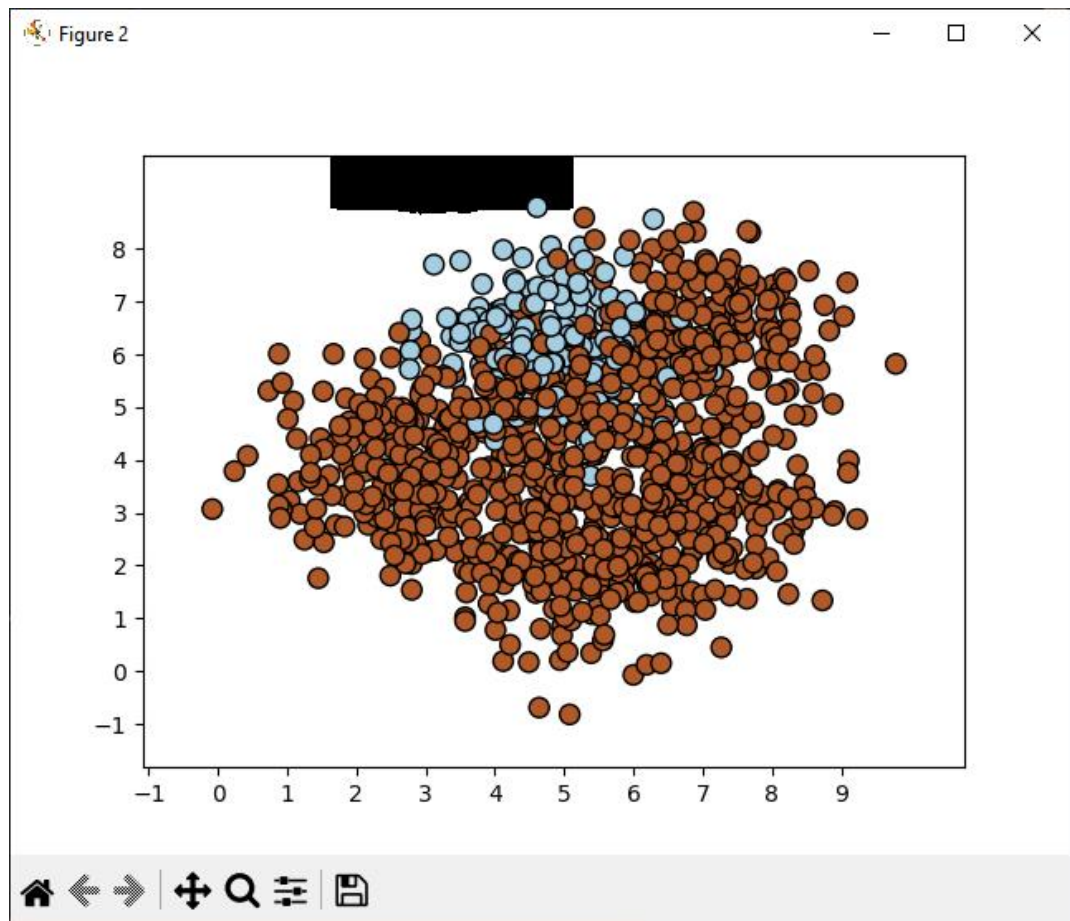


Рис.4.10 – Розподілення незбалансованих даних

```
#####
Classifier performance on training 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

#####
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.4.11 – Характеристика незбалансованого класифікатора

		Чижмотря М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

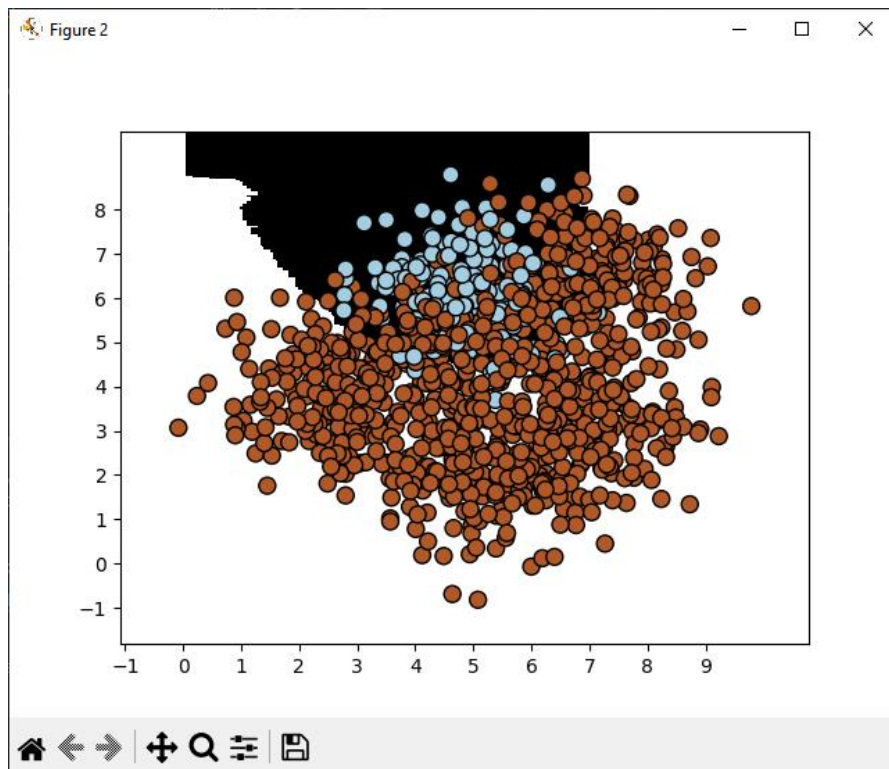


Рис.4.12 – Збалансована класифікація

```
#####
Classifier performance on training 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

#####
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.4.13 – Характеристики збалансованої класифікації

Завдяки балансуванню даних було отримано коректні результати та ефективно класифіковано дані

		Чижморя М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Лістинг коду файлу Task_3.py:

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

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("#### Searching optimal parameters for", metric)
    classifier = GridSearchCV(ExtraTreesClassifier(random_state=0),
parameter_grid, cv=5, scoring=metric)
    classifier.fit(X_train, Y_train)
    print("\nScores across the parameter grid:")

    for params, avg_score in classifier.cv_results_.items():
        print(params, '-->', avg_score)
    print("\nHighest scoring parameter set:", classifier.best_params_)

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

visualize_classifier(classifier, X_test, Y_test)
```

		Чижмотря М.О.			ДУ «Житомирська політехніка».19.121.2.000 - Лр4	Арк.
		Пулеко І.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Scores across the parameter grid:
mean_fit_time --> [0.13986268 0.12857866 0.13555651 0.14360156 0.15049534 0.0281621
 0.05654421 0.11057539 0.27175169]
std_fit_time --> [0.01949191 0.02776953 0.01772089 0.01587426 0.01854836 0.00206679
 0.00559401 0.00425106 0.00582361]
mean_score_time --> [0.01545153 0.01508284 0.01636391 0.01521058 0.01745324 0.00403328
 0.00978813 0.01174321 0.02912688]
std_score_time --> [0.00301622 0.00448382 0.0036678 0.00400837 0.00477281 0.00063775
```

Рис.4.14 – Отримання даних класифікації

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

   Class-0       0.94      0.81      0.87        79
   Class-1       0.81      0.86      0.83        70
   Class-2       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.4.15 – Характеристика класифікації зі сітковим пошуком

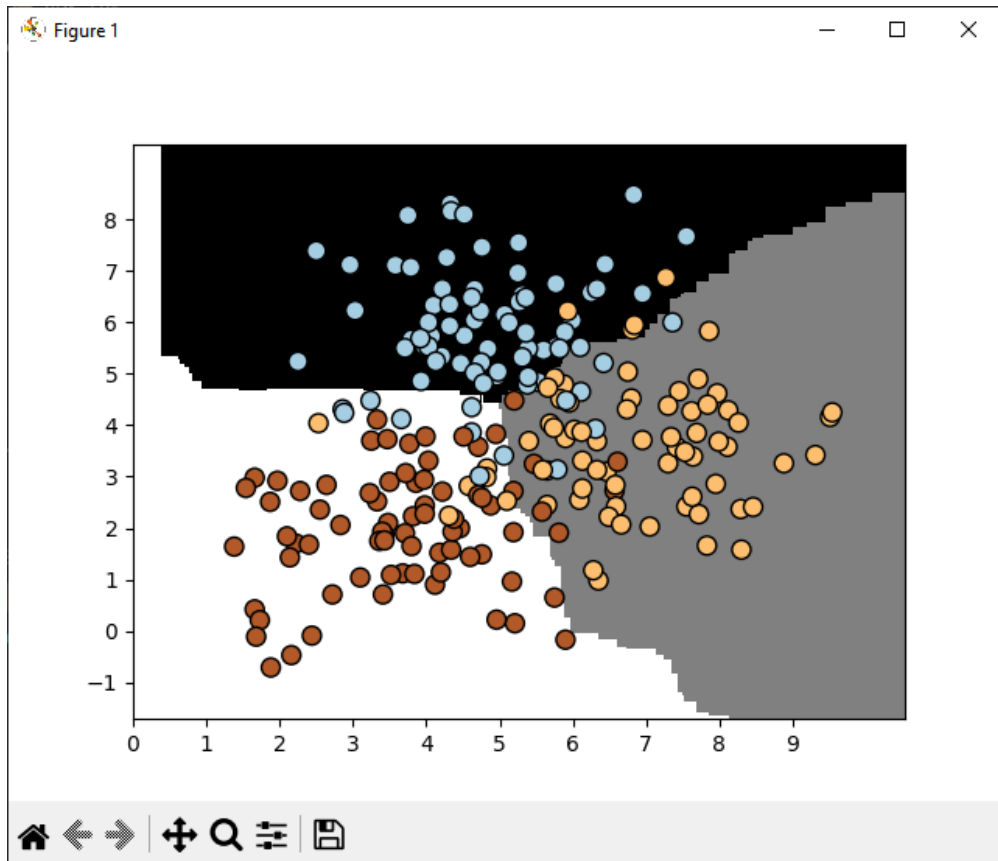


Рис.4.16 – Класифікація даних зі сітковим пошуком

Завдання 4. Обчислення відносної важливості ознак

Виконання завдання неможливе, дані є застарілими та доступ до них обмежено.

Завдання 5. Прогнозування інтенсивності дорожнього руху за допомогою класифікатора на основі гранично випадкових лісів

Лістинг коду Task_5.py:

```
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, cross_val_score
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)

label_encoder = []
X_encoded = np.empty(data.shape)
```

		Чижморя М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулко І.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

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

```

Mean absolute error = 7.42
Predicted traffic: 26

Рис.4.17 – Результат регресії на основі гранично випадкових лісів

Завдання 6. Створення навчального конвеєра (конвеєра машинного навчання)

Лістинг файлу Task_6.py:

```

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=10)
classifier = ExtraTreesClassifier(n_estimators=60, max_depth=4)

pipeline = Pipeline([('selector', k_best_selector), ('erf', classifier)])
pipeline.set_params(selector__k=7, erf__n_estimators=30)
pipeline.fit(X, Y)

```

		Чижморя М.О.			ДУ «Житомирська політехніка».19.121.2.000 - Лр4	Арк.
		Пулко І.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```

print("Predicted output:", pipeline.predict(X))

print("Score:", pipeline.score(X, Y))
status = pipeline.named_steps['selector'].get_support()
selected = [i for i, x in enumerate(status) if x]
print("Selected features:", selected)

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

Score: 0.86

Selected features: [4, 7, 8, 12, 14, 17, 22]

```

Рис.4.18 – Отримані результати навчального конвеєра

Завдання 7. Пошук найближчих сусідів

Лістинг файлу Task_7.py:

```

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_data = np.array([[4.3, 2.7]])

knn = NearestNeighbors(n_neighbors=k, algorithm='ball_tree').fit(X)
distances, indices = knn.kneighbors(test_data)

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

plt.figure()
plt.title("K Nearest Neighbors")
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='k')
plt.scatter(test_data[:, 0], test_data[:, 1], marker='o', s=75, color='red')
plt.scatter(X[indices[0][0][:k], 0], X[indices[0][0][:k], 1], marker='o', s=250,
            color='k', facecolors='none')
plt.show()

```

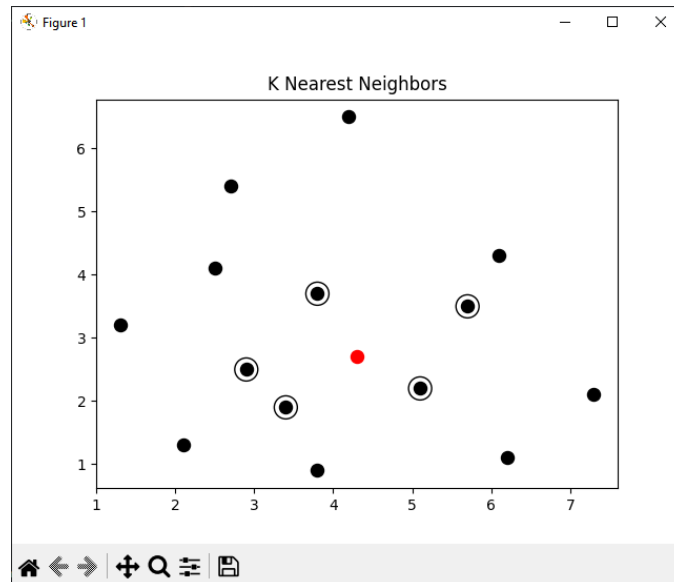


Рис.4.19 – Пошук найближчих сусідів

```
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.4.20 – Дані про найближчих сусідів

Завдання 8: Створити класифікатор методом k найближчих сусідів

Лістинг коду файлу Task_8.py:

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

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

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_mesh = classifier.predict(np.c_[X_values.ravel(), Y_values.ravel()])
output_mesh = output_mesh.reshape(X_values.shape)

plt.figure()
plt.pcolormesh(X_values, Y_values, output_mesh, cmap=cm.Paired)
plt.scatter(X[:, 0], X[:, 1], c=Y, s=80, edgecolors='black', linewidth=1,
```

```

cmap=cm.Paired)
plt.xlim(X_values.min(), X_values.max())
plt.ylim(Y_values.min(), Y_values.max())
plt.title('K Nearest Neighbors classifier on input data')

test_datapoint = [5.1, 3.6]
plt.scatter(test_datapoint[0], test_datapoint[1], marker='o', s=100, linewidths=3,
color='black')

_, indices = classifier.kneighbors([test_datapoint])
indices = np.asarray(indices).flatten()
plt.scatter(X[indices][:, 0], X[indices][:, 1], marker='*', s=80, linewidths=1,
color='black', facecolors='none')
plt.show()

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

```

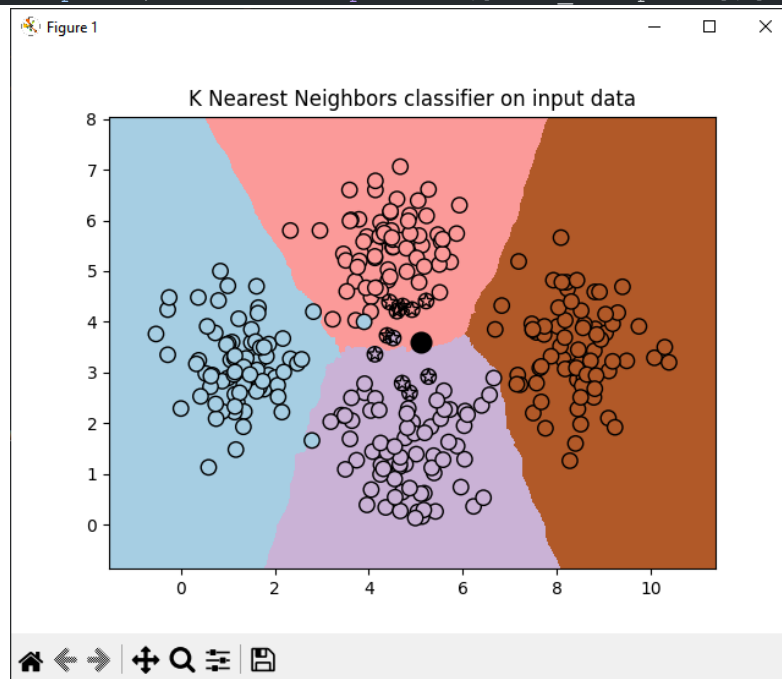


Рис.4.21 – Класифікація методом К-найближчих сусідів та найближчі сусіди введеної точки

Predicted output: 1.0

Рис.4.22 – Обрахований клас точки

Завдання 9. Обчислення оцінок подібності

Лістинг коду файлу Task_9.py:

```

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")

```

		Чижмотря М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		


```

        parser.add_argument('--score-type', dest='score_type', required=True,
                             choices=['Euclidean', 'Pearson'], help="Similarity score
to be computed")
        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

    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])

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

    Sxy = product_sum - (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)

```

		Чижмостря М.О.			ДУ «Житомирська політехніка».19.121.2.000 - Лр4	Арк.
		Пулеко І.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

    with open('movie_ratings.json', '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))

```

```

(venv) D:\LRs\AI\LR4>python Task_9.py --user1 "David Smith" --user2 "Bill Duffy" --score-type Euclidean

Euclidean score:
0.585786437626905

(venv) D:\LRs\AI\LR4>python Task_9.py --user1 "David Smith" --user2 "Bill Duffy" --score-type Pearson

Pearson score:
0.9909924304103233

(venv) D:\LRs\AI\LR4>

```

Рис.4.23 – Обрахунок оцінок для David Smith та Bill Duffy

```

(venv) D:\LRs\AI\LR4>python Task_9.py --user1 "David Smith" --user2 "Brenda Peterson" --score-type Pearson

Pearson score:
-0.7236759610155113

(venv) D:\LRs\AI\LR4>python Task_9.py --user1 "David Smith" --user2 "Brenda Peterson" --score-type Euclidean

Euclidean score:
0.1424339656566283

(venv) D:\LRs\AI\LR4>

```

Рис.4.24 – Обрахунок оцінок для David Smith та Brenda Peterson

		Чижмотря М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```
(venv) D:\LRs\AI\LR4>python Task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Euclidean
Euclidean score:
0.30383243470068705

(venv) D:\LRs\AI\LR4>python Task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Pearson
Pearson score:
0.7587869106393281

(venv) D:\LRs\AI\LR4>
```

Рис.4.25 – Обрахунок оцінок для David Smith та Samuel Miller

Завдання 10. Пошук користувачів зі схожими уподобаннями методом колаборативної фільтрації

Лістинг файлу Task_10.py:

```
import argparse
import json
import numpy as np
from 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]
    return scores[top_users]

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

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

    print("Users similar to " + user + ":")
    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))
```

		Чижмоторя М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```
(venv) D:\LRs\AI\LR4>python 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

(venv) D:\LRs\AI\LR4>
```

Рис.4.26 – Знаходження найбільших оцінок

Завдання 11. Створення рекомендаційної системи фільмів

Лістинг файлу Task_11.py:

```
import argparse
import json
import numpy as np
from Task_9 import pearson_score
from Task_10 import find_similar_users

def build_arg_parser():
    parser = argparse.ArgumentParser(description='Find movies recommended for the input 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')

    total_scores = {}
    similarity_sums = {}
    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 = [movie for movie in dataset[user]
                        if movie not in dataset[input_user] or
dataset[input_user][movie] == 0]

        for movie in filtered_list:
            total_scores.update({movie: dataset[user][movie] * similarity_score})
            similarity_sums.update({movie: similarity_score})

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

    movie_ranks = np.array([[total/similarity_sums[item], item] for item, total in
total_scores.items()])
    movie_ranks = movie_ranks[np.argsort(movie_ranks[:, 0])[:, -1]]
    recommended_movies = [movie for _, movie in movie_ranks]
```

		Чижмоторя М.О.			ДУ «Житомирська політехніка».19.121.2.000 - Лр4	Арк.
		Пулеко І.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return recommended_movies[:10]

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

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

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

```

```

(venv) D:\LRs\AI\LR4>python Task_11.py --user "Chris Duncan"
Movies recommended for Chris Duncan:
1. Vertigo
2. Scarface
3. Goodfellas
4. Roman Holiday

```

Рис.4.27 – Результат пошуку рекомендацій

Висновок: під час виконання завдань лабораторної роботи було досліджено методи ансамблів у машинному навчанні та створено рекомендаційні системи використовуючи спеціалізовані бібліотеки та мову програмування Python.

Github: https://github.com/mikrorobot/Python_AI

		Чижмоторя М.О.			ДУ «Житомирська політехніка».19.121.2.000 – Лр4	Арк.
		Пулеко І.В.				21
Змн.	Арк.	№ докум.	Підпис	Дата		