

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

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

# 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».22.121.10.000 – Лр4			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи			
Розроб.		Кочубей К.М.						
Перевір.		Голенко М. Ю.						
Керівник								
Н. контр.								
Зав. каф.					ФІКТ Гр. ІПЗ-20-1[1]			
					Лім.	Арк.	Аркушів	
						1	23	

```

# 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()

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

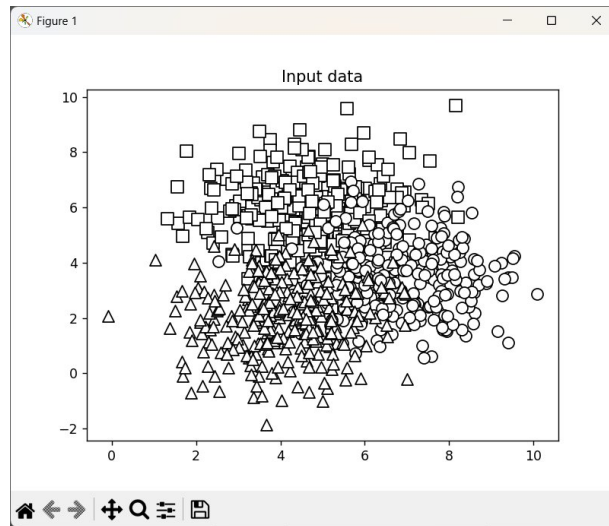


Рис.1. Результат виконання LR\_4\_task\_1.py

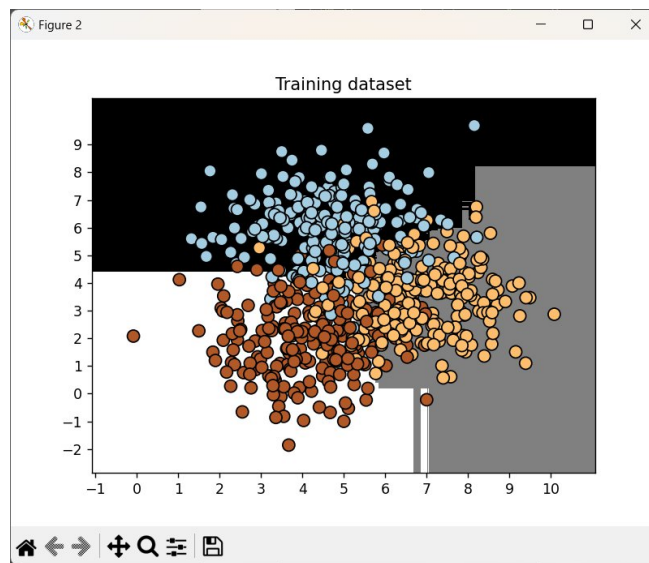


Рис.2. Результат виконання LR\_4\_task\_1.py

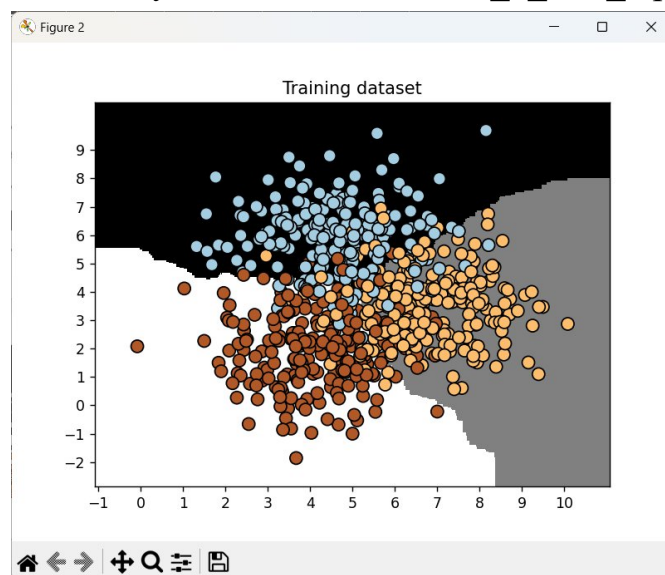


Рис.3. Результат виконання LR\_4\_task\_1.py

## Завдання 2.2. 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()
```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

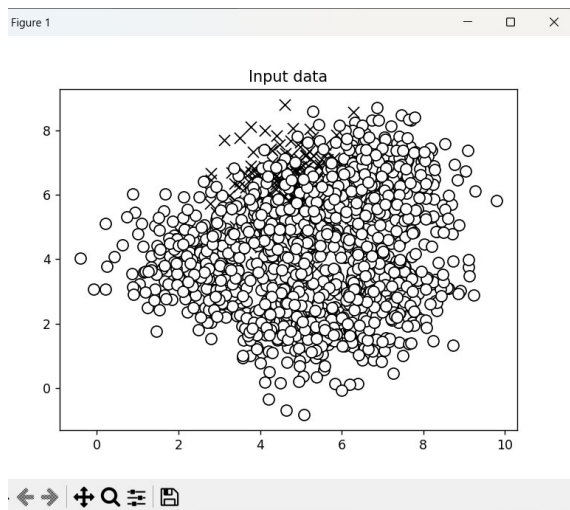


Рис.4. Результат виконання LR\_4\_task\_2.py

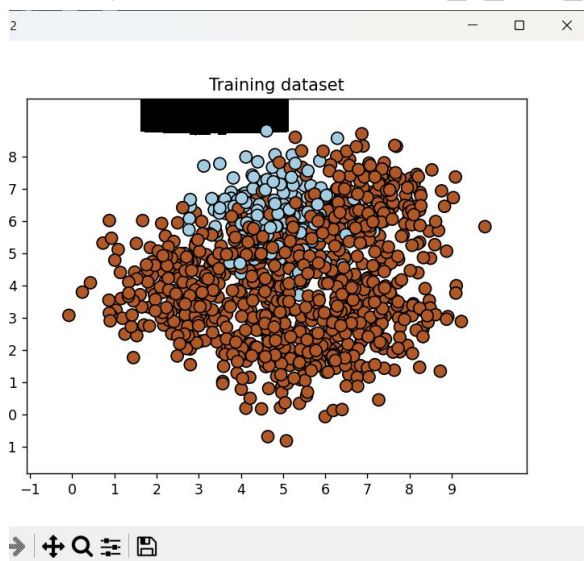


Рис.5. Результат виконання LR\_4\_task\_2.py

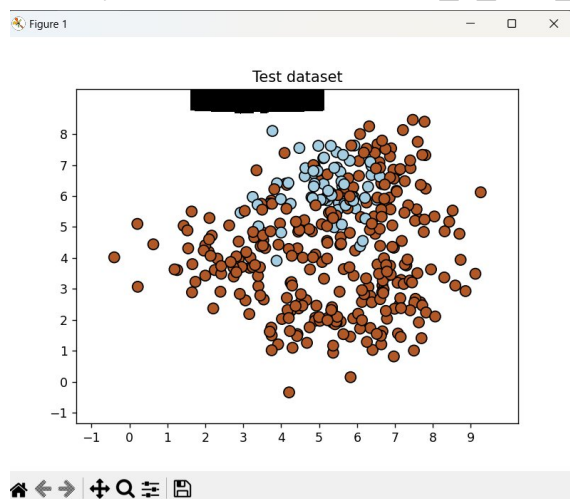


Рис.6. Результат виконання LR\_4\_task\_2.py

```
#####

Classifier performance on training dataset

C:\Program Files\Python310\lib\site-packages\sklearn\metr
_warn_prf(average, modifier, msg_start, len(result))
          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

C:\Program Files\Python310\lib\site-packages\sklearn\metr
_warn_prf(average, modifier, msg_start, len(result))
C:\Program Files\Python310\lib\site-packages\sklearn\metr
_warn_prf(average, modifier, msg_start, len(result))
          precision    recall  f1-score   support

   Class-0         0.00      0.00      0.00         69
   Class-1         0.82      1.00      0.90        306

 accuracy          0.82        375
 macro avg         0.41      0.50      0.45        375
weighted avg         0.67      0.82      0.73        375

#####
```

Рис.7. Результат виконання LR\_4\_task\_2.py

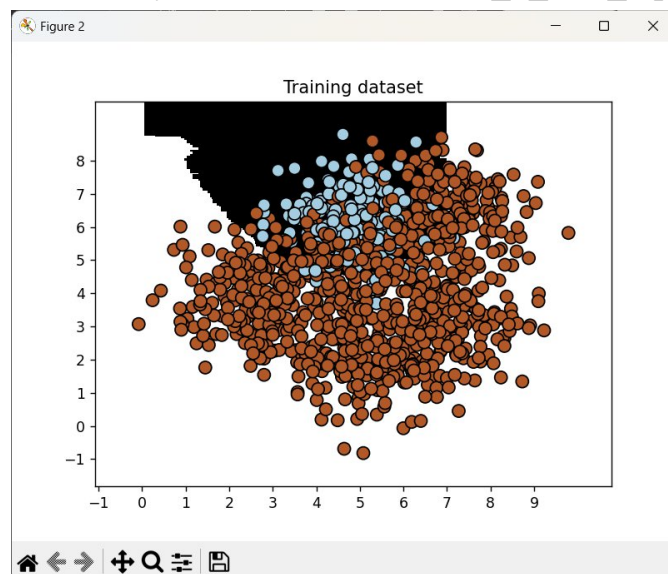


Рис.8. Результат виконання LR\_4\_task\_2.py



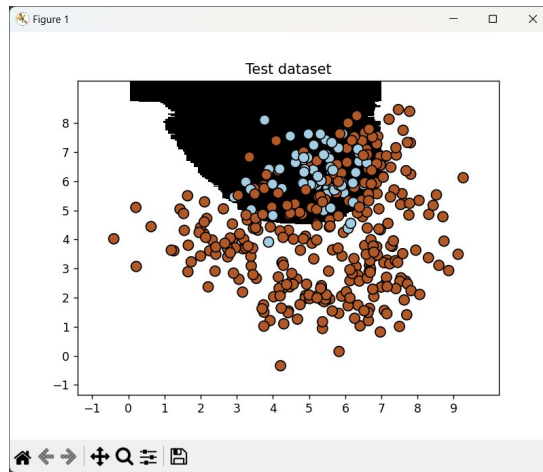


Рис.9. Результат виконання LR\_4\_task\_2.py

```
#####

      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.10. Результат виконання LR\_4\_task\_2.py

### Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою 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
from sklearn.metrics import classification_report
```

```

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))

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		8



```
##### Searching optimal parameters for precision_weighted
mean_score_time  std_score_time  ... std_test_score rank_test_score
0      0.006898      0.001481  ...      0.025132          1
1      0.005814      0.000245  ...      0.023032          5
2      0.006104      0.000201  ...      0.026560          4
3      0.006841      0.000517  ...      0.028334          8
4      0.007552      0.000389  ...      0.033429          9
5      0.002806      0.000242  ...      0.029698          2
6      0.004111      0.000206  ...      0.020401          7
7      0.006122      0.000484  ...      0.023032          5
8      0.012839      0.000420  ...      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
macro avg          0.86        0.86        0.86         225
weighted avg          0.86        0.86        0.86         225
```

Рис.11. Результат виконання LR\_4\_task\_3.py

```
##### Searching optimal parameters for recall_weighted
mean_score_time  std_score_time  ... std_test_score rank_test_score
0      0.006409      0.000375  ...      0.027075          1
1      0.006710      0.000511  ...      0.022468          5
2      0.006708      0.000514  ...      0.026749          3
3      0.007768      0.000325  ...      0.028497          8
4      0.007823      0.000624  ...      0.034744          9
5      0.002606      0.000372  ...      0.029407          1
6      0.004115      0.000203  ...      0.020096          7
7      0.006535      0.000638  ...      0.022468          5
8      0.012933      0.000661  ...      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
macro avg          0.86        0.86        0.86         225
weighted avg          0.86        0.86        0.86         225
```

Рис.11. Результат виконання LR\_4\_task\_3.py

## Завдання 2.4. Обчислення відносної важливості ознак

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
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
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
feature_names = np.array(["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS",
                          "RAD", "TAX", "PTRATIO", "B", "LSTAT", "MEDV"])
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

# Перемішування даних
X, y = shuffle(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_

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

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

# Розміщення міток уздовж осі X
pos = np.arange(index_sorted.shape[0]) + 0.5
# Побудова стовпчастої діаграми
plt.figure()
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, feature_names[index_sorted])
plt.ylabel('Relative Importance')
plt.title('Оцінка важливості ознак з використанням регресора AdaBoost')
plt.show()

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

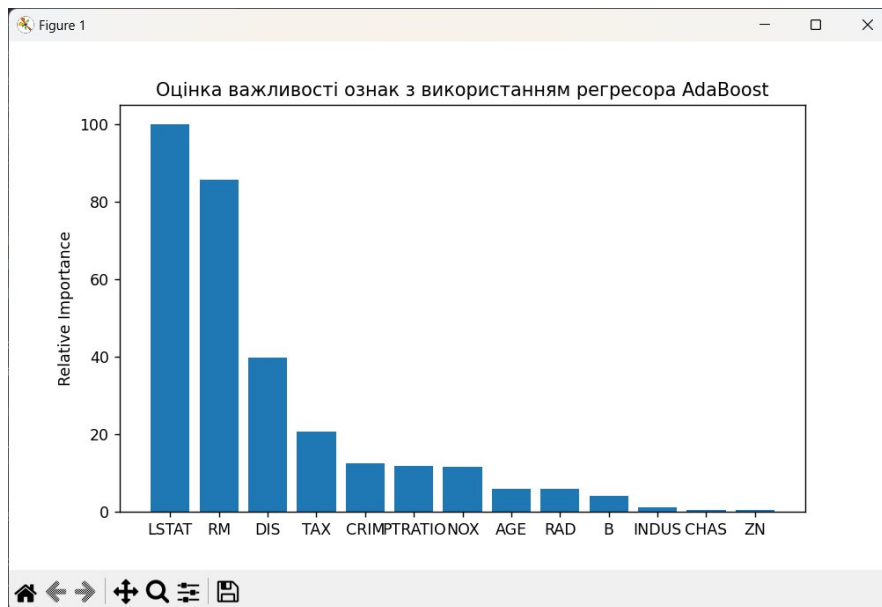


Рис.12. Результат виконання LR\_4\_task\_4.py

```
ADABOOST REGRESSOR
Mean squared error = 22.7
Explained variance score = 0.79
```

Рис.13. Результат виконання LR\_4\_task\_4.py

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

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, mean_absolute_error
from sklearn 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])
```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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]))

```

```

Mean absolute error: 7.42
Predicted traffic: 26

```

Рис.14. Лінійна регресія LR\_4\_task\_5.py

## Завдання 2.6. 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)])

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Пр4	Арк.
		Голенко М. Ю.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# 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]))
```

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

Score: 0.8933333333333333

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

Рис.15. Результати виконання LR\_4\_task\_6.py

У першому списку містяться спрогнозовані вихідні мітки.

Значення Score означає рівень ефєктивності конвеєра.

В останньому рядку містяться індекси ознак.

## Завдання 2.7. 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')
```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# 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])

# 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][: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()
```

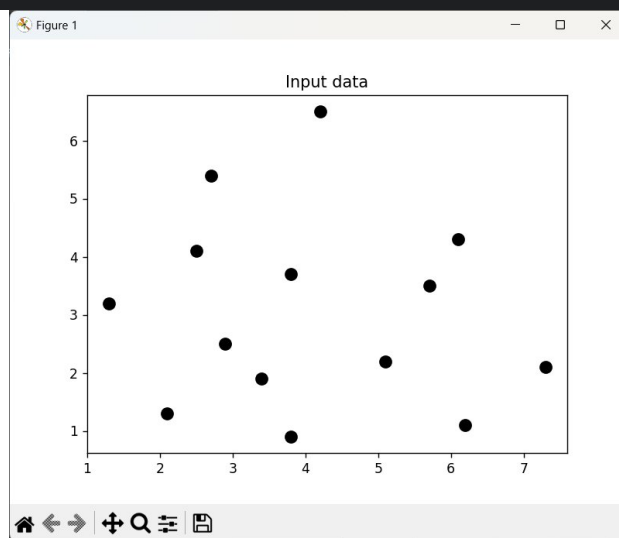


Рис.16. Результати виконання LR\_4\_task\_7.py

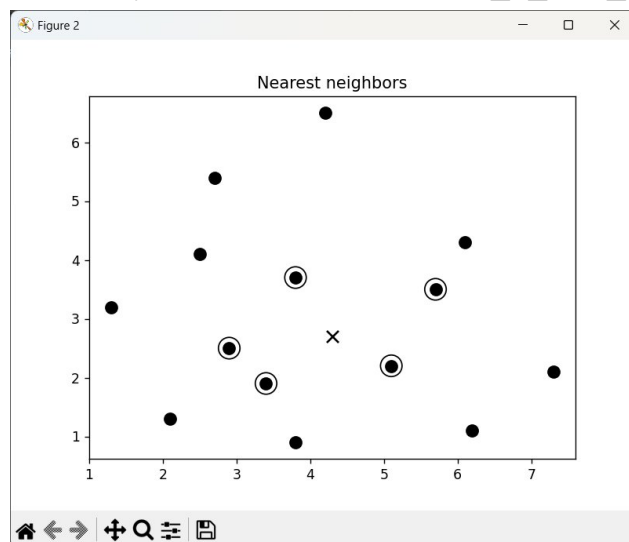


Рис.17. Результати виконання LR\_4\_task\_7.py



```
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.18. Результати виконання LR\_4\_task\_7.py

### Завдання 2.8. 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(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)

# 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')
```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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(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()

```

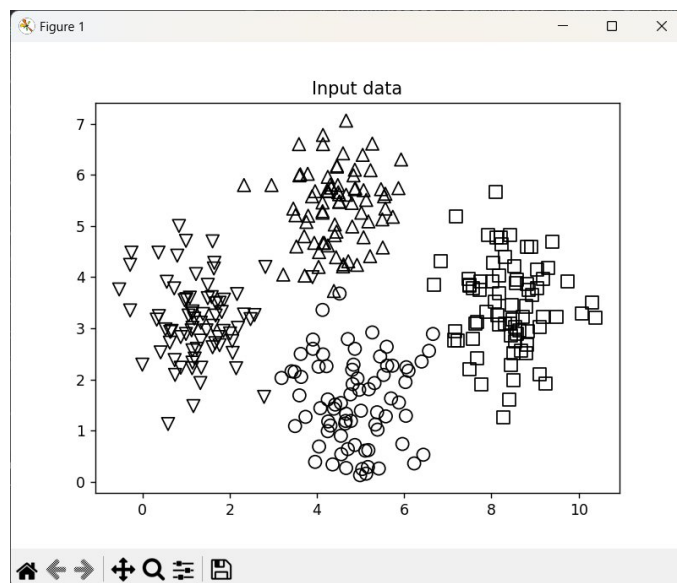


Рис.19. Вхідні дані LR\_4\_task\_8.py

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

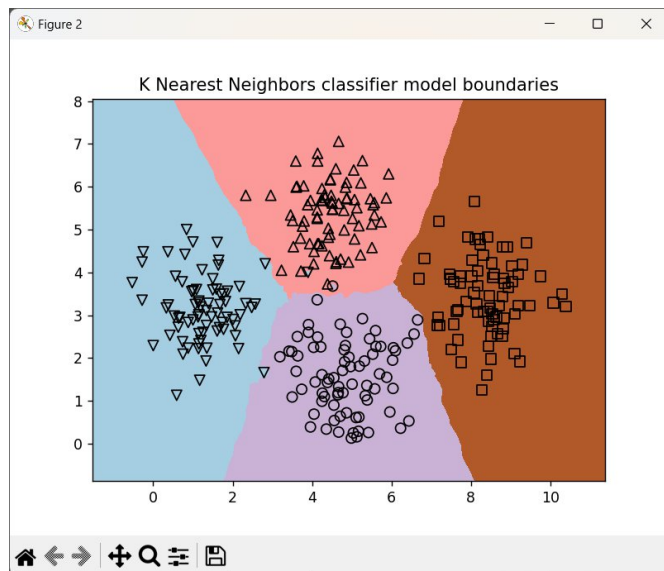


Рис.20. Межі класифікатора LR\_4\_task\_8.py

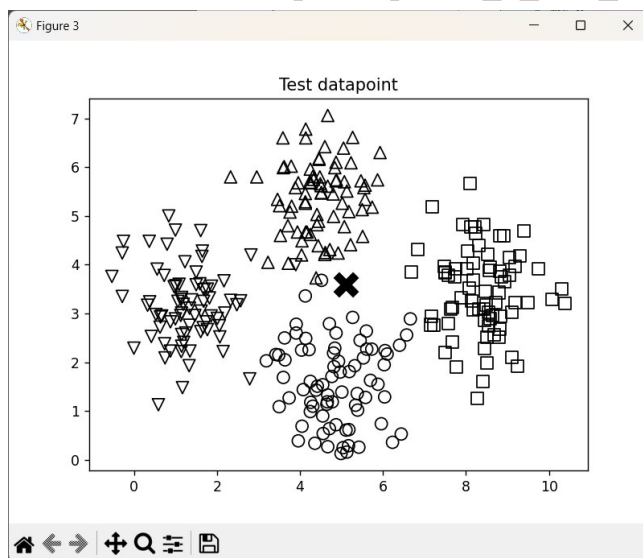


Рис.21. Тестова точка LR\_4\_task\_8.py

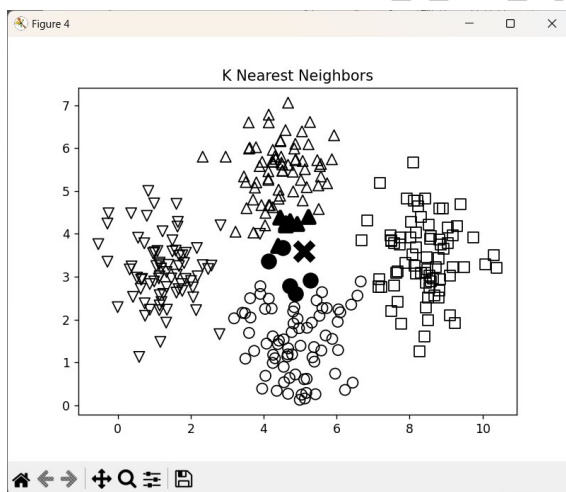


Рис.22. Найближчі сусіди LR\_4\_task\_8.py

Predicted output: 1

Рис.23. Результати виконання LR\_4\_task\_8.py

### Завдання 2.9. 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')
    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)
```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# 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 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))

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Bill Duffy" --score-type Euclidean

Euclidean score:
0.585786437626905
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Bill Duffy" --score-type Pearson

Pearson score:
0.9909924304103233
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Brenda Peterson" --score-type Euclidean

Euclidean score:
0.1424339656566283
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Brenda Peterson" --score-type Pearson

Pearson score:
-0.7236759610155113
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Euclidean

Euclidean score:
0.30383243470068705
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Pearson

Pearson score:
0.7587869106393281
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Euclidean

Euclidean score:
0.2857142857142857
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Pearson

Pearson score:
0
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Chris Duncan" --score-type Euclidean

Euclidean score:
0.38742588672279304
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_9.py --user1 "David Smith" --user2 "Chris Duncan" --score-type Pearson

Pearson score:
1.0

```

Рис.24. Результати виконання LR\_4\_task\_9.py

## Завдання 2.10. Пошук користувачів зі схожими уподобаннями методом колаборативної 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')

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				20
Змн.	Арк.	№ докум.	Підпис	Дата		



```

# 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])

# Сортуювання 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]

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))

```

```

PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 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
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 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.25. Результати виконання LR\_4\_task\_10.py

## Завдання 2.11. 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

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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()])

    # Сортування 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)

```

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_11.py --user "Chris Duncan"

Movie recommendations for Chris Duncan:
1. Vertigo
2. Scarface
3. Goodfellas
4. Roman Holiday
PS C:\Study\4 курс 1 семестр\AI\Lab4> python3 LR_4_task_11.py --user "Julie Hammel"

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

```

Рис.25. Результати виконання LR\_4\_task\_11.py

Репозиторій: [https://github.com/Kochubei-Kostiantyn/AI\\_labs](https://github.com/Kochubei-Kostiantyn/AI_labs)

**Висновки:** в ході виконання лабораторної роботи ми використовуємо спеціалізовані бібліотеки та мову програмування Python дослідили методи ансамблів у машинному навчанні та створили рекомендаційні системи.

		Кочубей К. М.			ДУ «Житомирська політехніка».22.121.10.000 – Лр4	Арк.
		Голенко М. Ю.				23
Змн.	Арк.	№ докум.	Підпис	Дата		