

ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ ТА СТВО- РЕННЯ 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

Лістинг програми:

```
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)
```

					ДУ «Житомирська політехніка».23.121.05.000 – Лр4			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи	Лім.	Арк.	Аркушів
Розроб.		Дубинченко Б.М.					1	23
Перевір.		Іванов Д.А.				ФІКТ Гр. ІПЗ-20-4[1]		
Керівник								
Н. контр.								
Зав. каф.								

```

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='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")

```

Результат виконання програми:

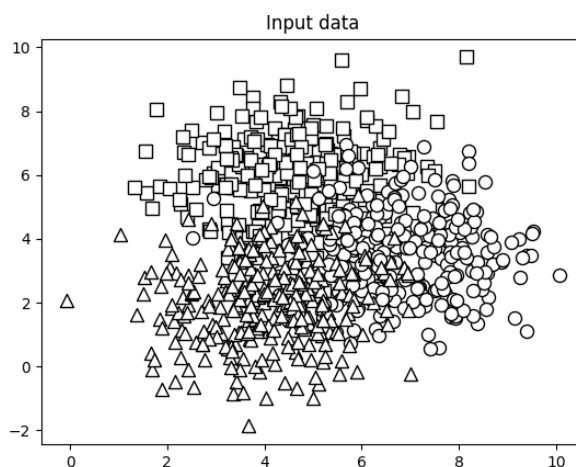


Рис. 4.1 Зображення розподілення даних

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

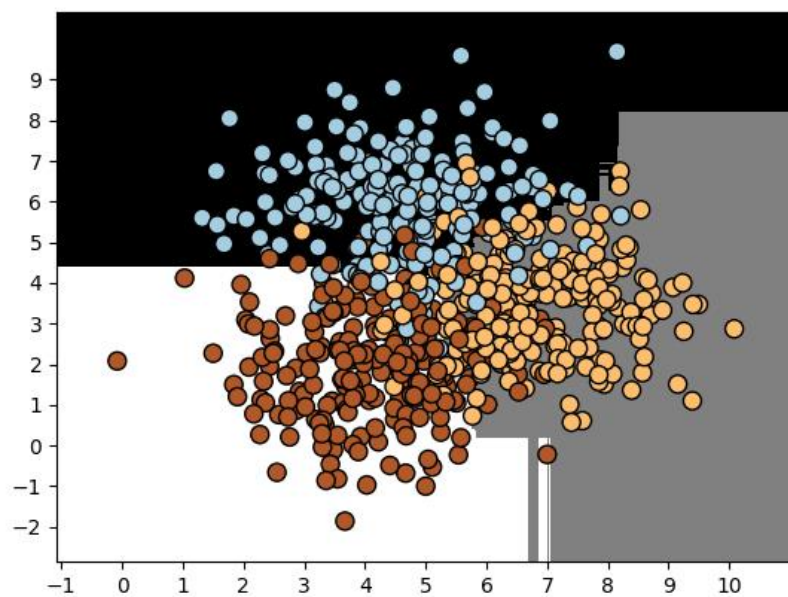


Рис. 4.2 Класифікація методом випадкових дерев

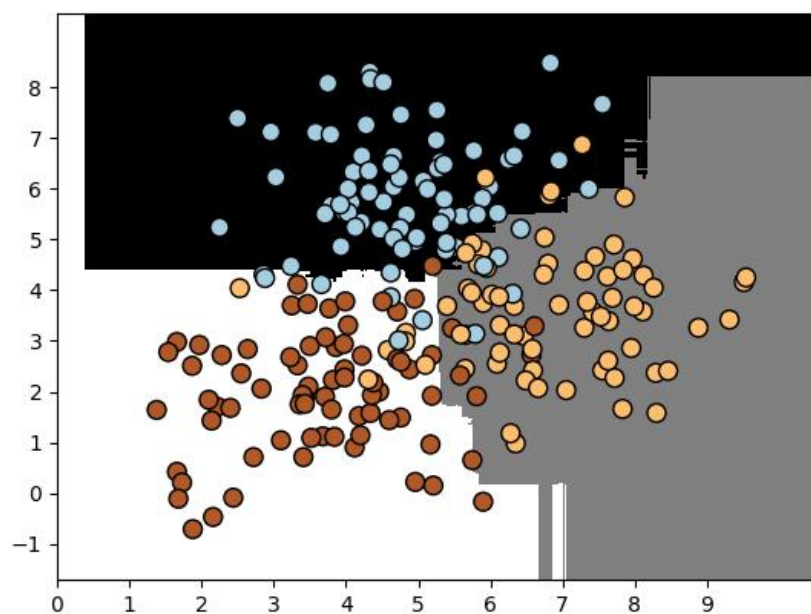


Рис. 4.3 Класифікація методом випадкових дерев

```
#####

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. 4.4 Характеристики роботи методу 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        0.87        0.87        225
weighted avg       0.87        0.87        0.87        225

#####
```

Рис. 4.5 Характеристики роботи методу гранично випадкових дерев

```

Confidence measure:

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

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

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

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

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

Datapoint: [5 2]
Predicted class: Class-2

```

Рис. 4.6. Дані про можливі класи (rf)

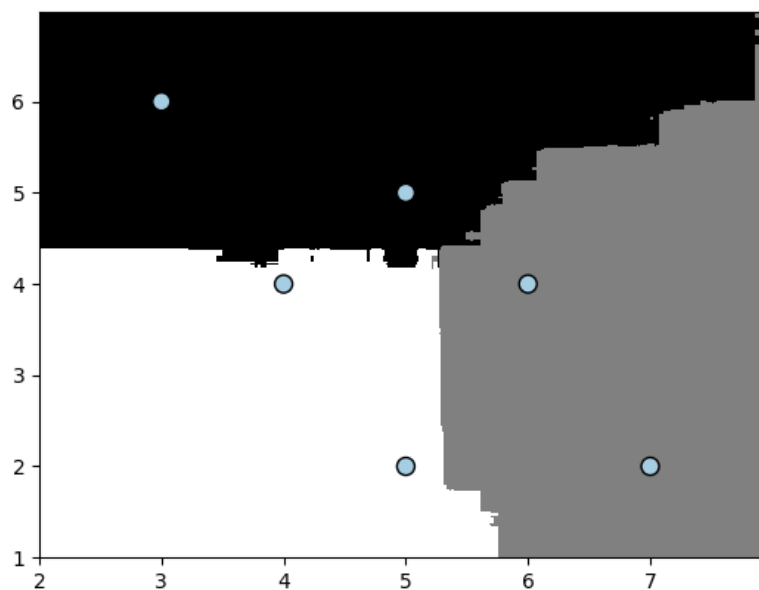


Рис. 4.7 Візуалізація можливих класів точок (rf).

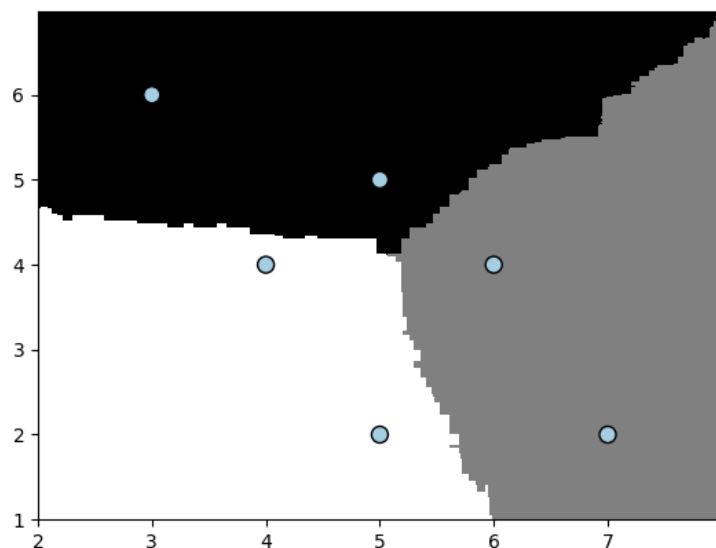


Рис 4.8 Візуалізація можливих класів точок (erf)

```
Confidence measure:

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

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

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

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

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

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

Рис 4.9 Дані про можливі класи (erf)

Завдання 2.2. Обробка дисбалансу класів

Лістинг програми:

```
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
```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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_test_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()

```

Результат виконання програми:

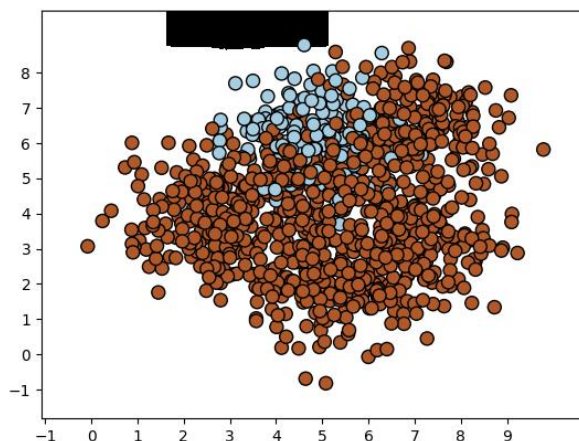


Рис 4.10 Розподілення незбалансованих даних

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Пр4	Арк.
		Іванов Д.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```
#####
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

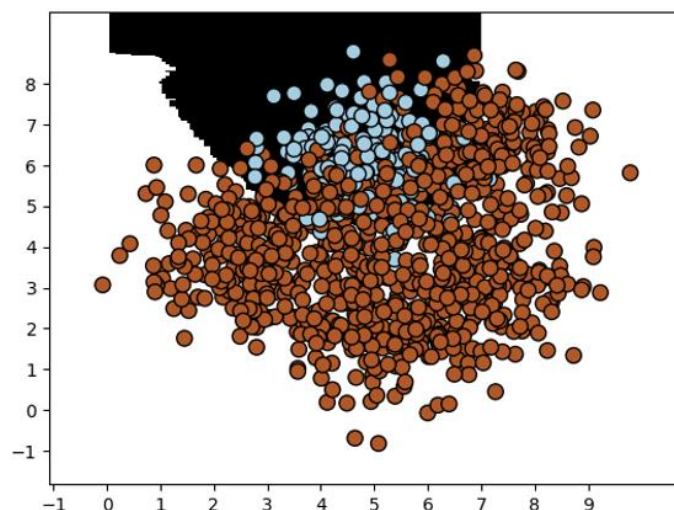


Рис 4.11 Збалансована класифікація

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 4.12 Характеристики збалансованої класифікації

Завдання 2.3. Знаходження оптимальних навчальних параметрів за допомогою сіткового пошуку

Лістинг програми:

Результат виконання програми:

```

### Searching optimal parameters for precision_weighted

Scores across the parameter grid:
mean_fit_time --> [0.06507735 0.06094617 0.07191281 0.08859296 0.09950008 0.01754003
0.03497458 0.07402754 0.16605668]
std_fit_time --> [0.00049361 0.00042755 0.00122083 0.00146205 0.00225427 0.0004445
0.00306372 0.01302652 0.0020101 ]
mean_score_time --> [0.00685902 0.00654473 0.00724034 0.0078136 0.00823941 0.00279641
0.00463338 0.00674338 0.01505647]
std_score_time --> [0.00041701 0.00032923 0.00041792 0.00062227 0.0002747 0.00039888
0.00060095 0.00021701 0.00039644]
param_max_depth --> [2 4 7 12 16 4 4 4]
param_n_estimators --> [100 100 100 100 100 100 25 50 100 250]
params --> [{ 'max_depth': 2, 'n_estimators': 100}, { 'max_depth': 4, 'n_estimators': 100}, { 'max_depth': 7, 'n_estimators': 100}, { 'max_depth': 12, 'n_estimators': 100}, { 'max_depth': 16, 'n_estimators': 100}, { 'max_depth': 4, 'n_estimators': 25}, { 'max_depth': 4, 'n_estimators': 50}, { 'max_depth': 4, 'n_estimators': 100}, { 'max_depth': 4, 'n_estimators': 250}]
split0_test_score --> [0.87406317 0.85323424 0.85381333 0.8154507 0.80037449 0.87558579
0.84512618 0.85323424 0.86067019]
split1_test_score --> [0.87696315 0.86737731 0.87662655 0.87651671 0.85190389 0.85594956
0.85035187 0.86737731 0.87897866]
split2_test_score --> [0.81956872 0.82834119 0.82611079 0.81030267 0.77569734 0.834651
0.83784535 0.82834119 0.82834119]
split3_test_score --> [0.82078374 0.80260075 0.80192593 0.80351421 0.7942149 0.7931046
0.80260075 0.80260075 0.80192593]
split4_test_score --> [0.8508842 0.85412387 0.86062409 0.85389639 0.86023157 0.86857693
0.86332922 0.85412387 0.85412387]
mean_test_score --> [0.8497566 0.84113547 0.84382014 0.83195501 0.81648444 0.84557357
0.83985067 0.84113547 0.84478797]
std_test_score --> [0.02513165 0.02303185 0.02050629 0.02833428 0.03342873 0.02969796
0.02040062 0.02303185 0.0268672 ]
rank_test_score --> [1 5 4 8 9 2 7 5 3]

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

```

Рис 4.13 Результат виконання завдання

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Пр4	Арк.
		Іванов Д.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Performance report:				
	precision	recall	f1-score	support
0.0	0.94	0.81	0.87	79
1.0	0.81	0.86	0.83	70
2.0	0.83	0.91	0.87	76
accuracy			0.86	225
macro avg	0.86	0.86	0.86	225
weighted avg	0.86	0.86	0.86	225

Рис 4.14 Характеристика класифікації зі сітковим пошуком

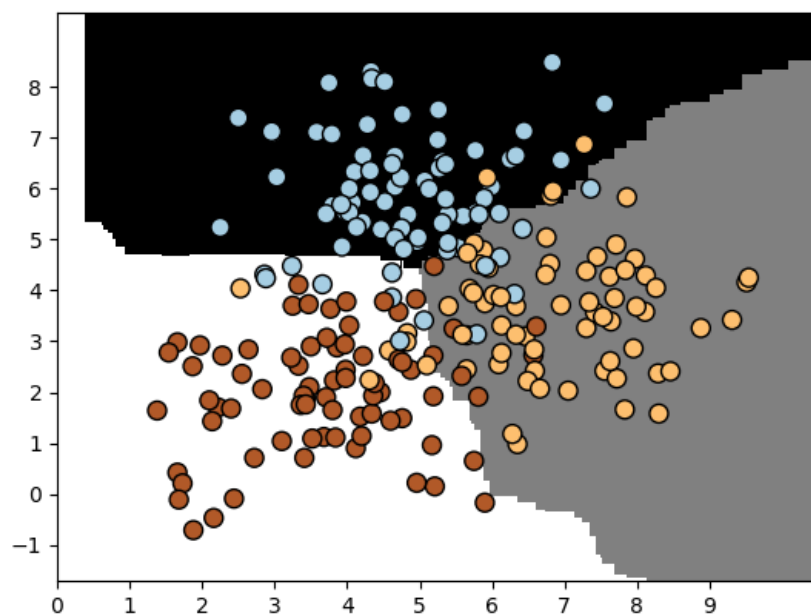


Рис 4.15 Класифікація даних зі сітковим пошуком

Завдання 2.4. Обчислення відносної важливості ознак

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
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.load_boston()
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)
```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

regressor = AdaBoostRegressor(DecisionTreeClassifier(max_depth=4),
n_estimators=400, random_state=7)
regressor.fit(X_train, Y_train)

Y_train_pred = regressor.predict(X_train)
mse = mean_squared_error(Y_train, Y_train_pred)
evs = explained_variance_score(Y_train, Y_train_pred)
print("ADABOOST REGRESSOR")
print("Mean squared error =", round(mse, 2))
print("Explained variance score =", round(evs, 2))

feature_importance = regressor.feature_importances_
feature_names = housing_data.feature_names

feature_importance = 100.0 * (feature_importance / max(feature_importance))

index_sorted = np.flipud(np.argsort(feature_importance))
pos = np.arange(index_sorted.shape[0]) + .5

plt.figure()
plt.bar(pos, feature_importance[index_sorted], align='center')
plt.xticks(pos, feature_names[index_sorted])
plt.ylabel('Relative Importance')
plt.title('Variable Importance')
plt.show()

```

Результат виконання завдання:

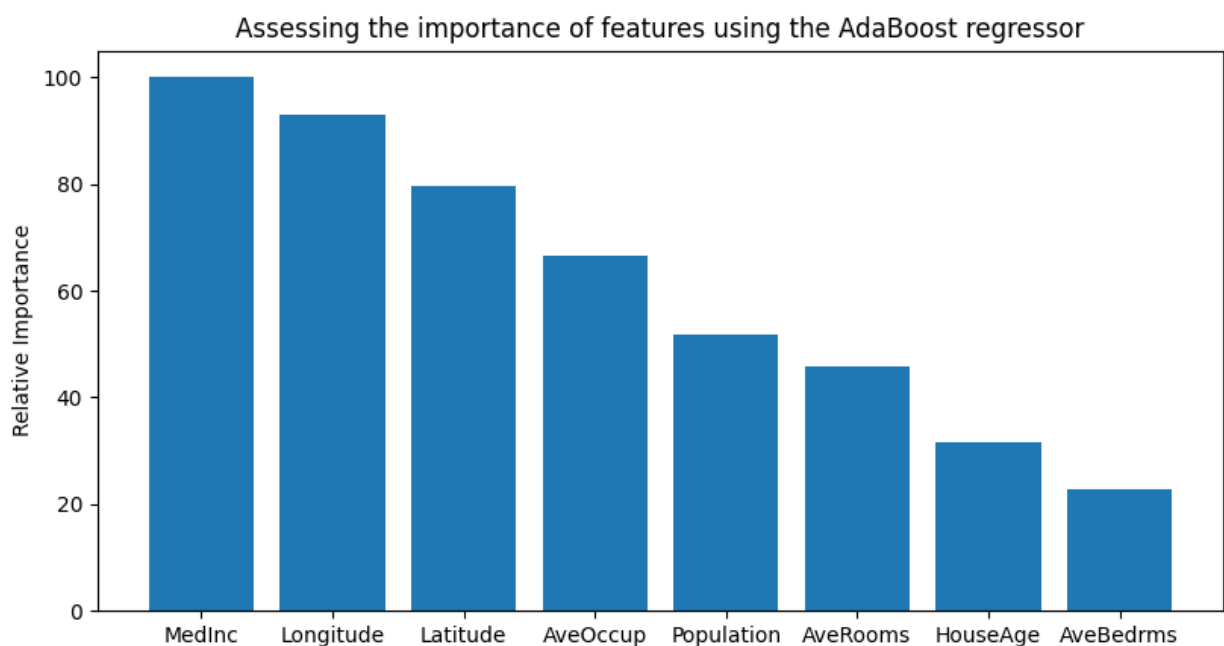


Рис 4.16 Обчислення відносної важливості ознак

```

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

```

Рис 4.17 Результат підрахунків

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

При виконання завдання, було помічено що дані є застарілими та немає до їх доступу.

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

Лістинг програми:

```
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)
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
```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Результат виконання завдання:

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

Process finished with exit code 0
```

Рис 4.18 Результат регресії на основі гранично випадкових лісів

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=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)

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

print("Score:", pipeline.score(X, Y))

status = 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 завдання:

```
Predicted output: [1 2 2 0 2 0 2 1 0 1 1 2 1 0 2 2 1 0 1 1 0 2 1 1 2 2 0 0 1 2 1 2 1 0 2 2 1
1 2 2 2 0 1 2 2 1 2 2 1 0 1 2 2 1 2 0 2 2 0 1 0 2 1 0 1 1 2 0 1 0 2
0 0 1 1 2 0 0 2 2 2 2 0 0 0 2 2 2 1 2 0 2 1 2 2 1 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 1 0 2 1 2 0 0 1 0 1 1 0 1 1 1 2 2 0 0 1 2 0
2 2]
Score: 0.8733333333333333

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

Рис 4.19 Отримані результати навчального конвеєра

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

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

plt.figure()
plt.title("Input data")
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='k')

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

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_datapoint[:, 0], test_datapoint[:, 1], marker='o', s=75,
            color='red')
plt.scatter(X[indices[0][0][:][:, 0], X[indices[0][0][:][:, 1], marker='o', s=250,
            color='k', facecolors='none')
plt.show()
```

Результат виконання завдання:

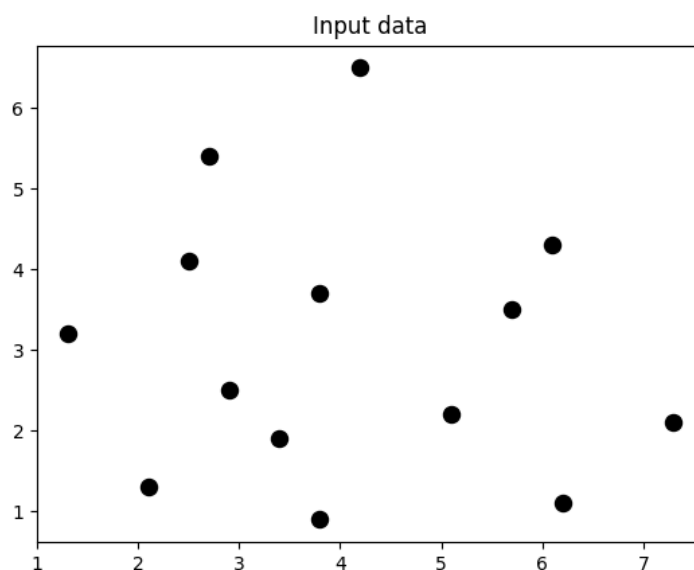


Рис 4.20 Вхідні данні

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

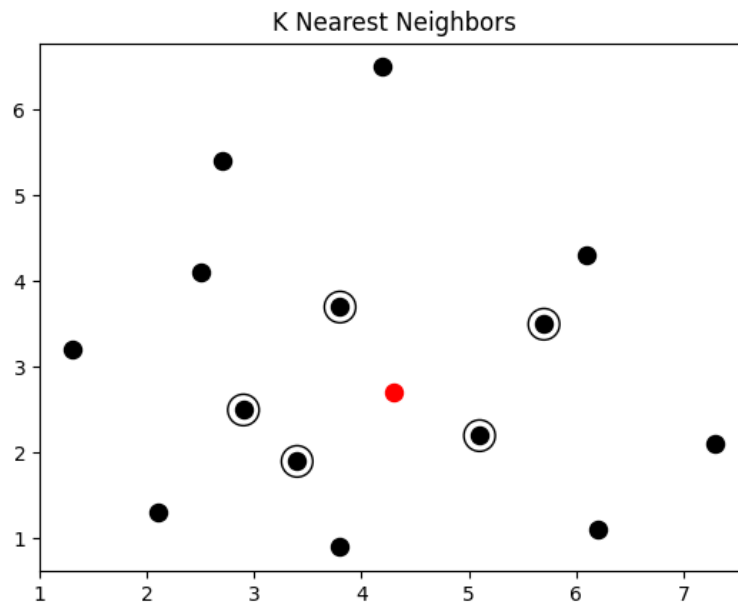


Рис 4.21 Пошук найближчих сусідів

```
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 4.22 Дані про 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, datasets

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

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_shapes[i], s=75, edgecolors='black',
                facecolors='none')

num_neighbors = 12
step_size = 0.01
classifier = neighbors.KNeighborsClassifier(num_neighbors, weights='distance')
```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Пр4	Арк.
		Іванов Д.А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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])

```

Результат виконання завдання:

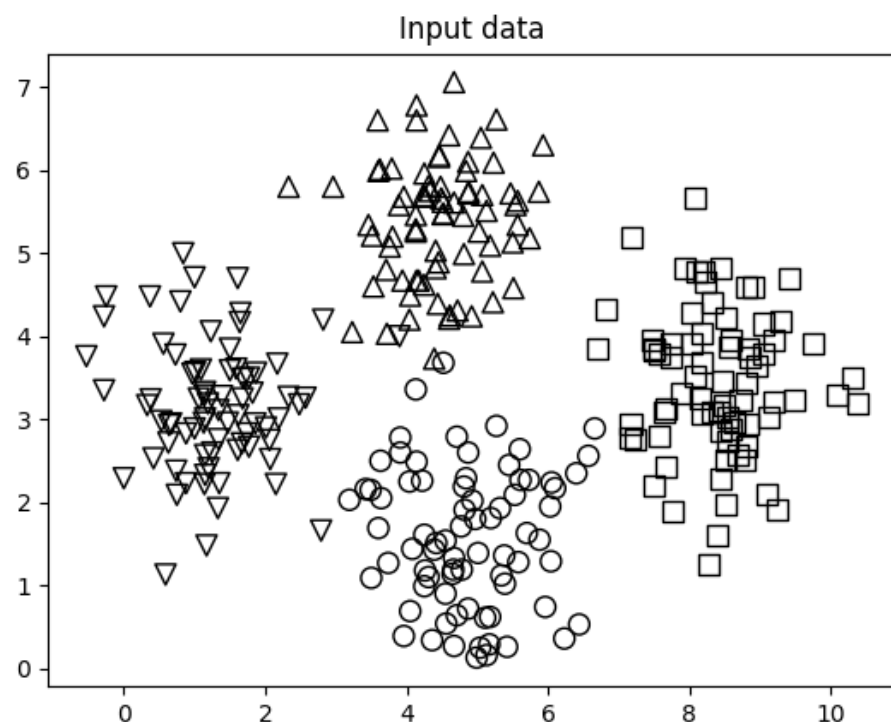


Рис 4.23 Вхідні дані

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

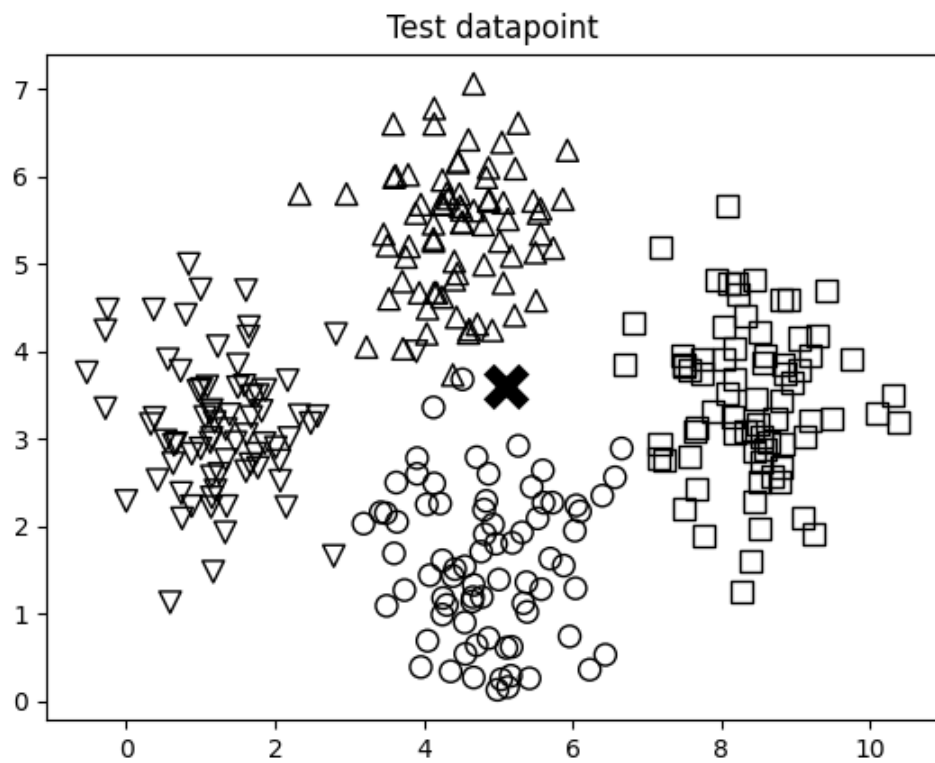


Рис 4.24 Тестова точка даних

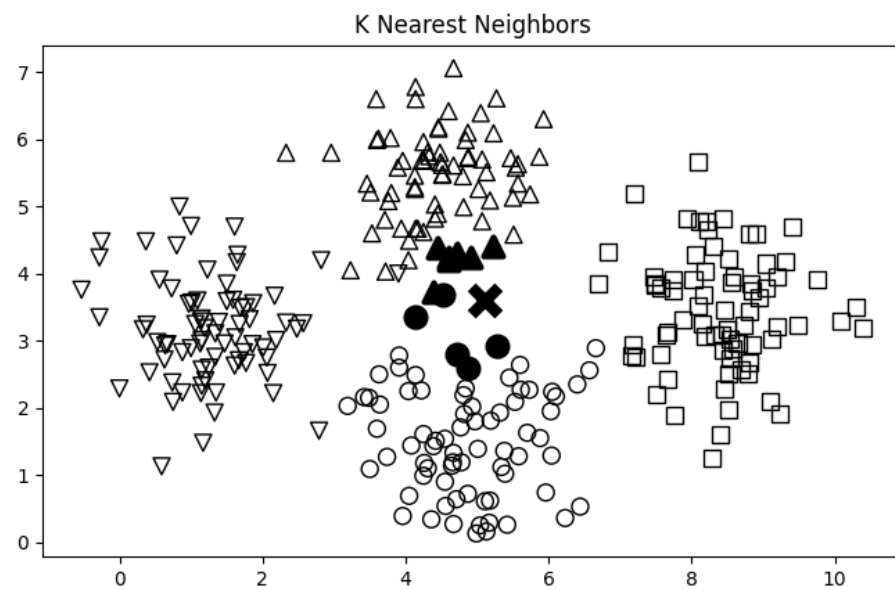


Рис 4.25 Найближчі сусіди

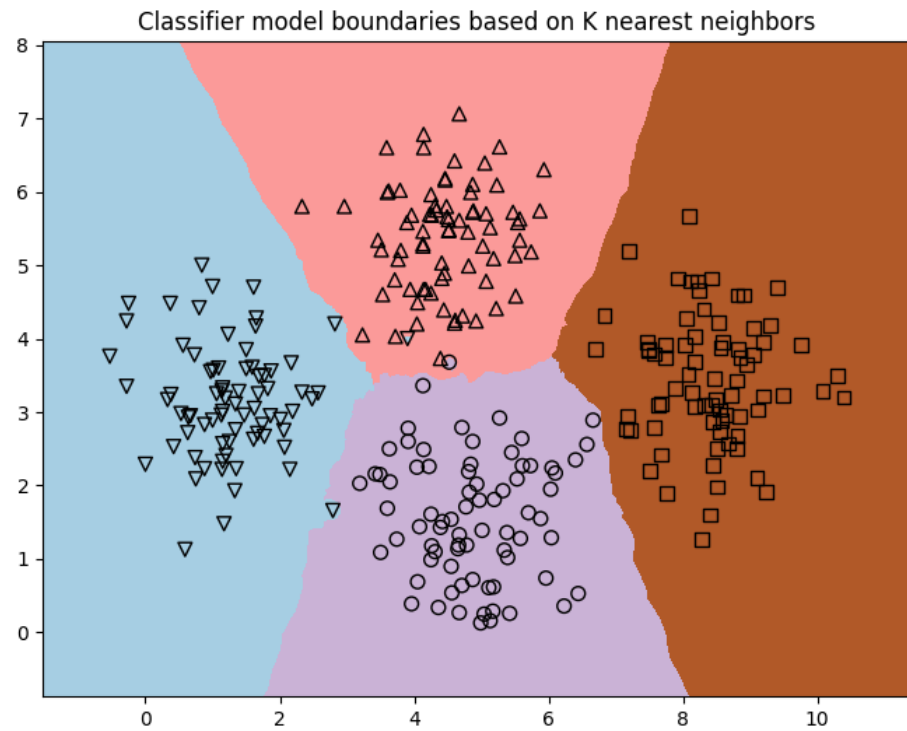


Рис 4.26 Класифікація методом К-найближчих сусідів та найближчі сусіди введеної точки

Predicted output: 1.0

Рис 4.27 Результат виконання програми

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 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 = {}
```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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)

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

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

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

```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```

else:
    print("\nPearson score:")
    print(pearson_score(data, user1, user2))

```

Результат виконання завдання:

```

(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Brenda Peterson" --score-type Euclidean

Euclidean score:
0.1424339656566283
(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Brenda Peterson" --score-type Pearson

Pearson score:
-0.7236759610155113
(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Samuel Miller" --score-type Euclidean

Euclidean score:
0.30383243470068705
(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Samuel Miller" --score-type Pearson

Pearson score:
0.7587869106393281
(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Julie Hammel" --score-type Euclidean

Euclidean score:
0.2857142857142857
(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Julie Hammel" --score-type Pearson

Pearson score:
0

```

Рис 4.28 Обрахунок оцінок

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

```
(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Clarissa Jackson" --score-type Euclidean

Euclidean score:
0.28989794855663564
(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Clarissa Jackson" --score-type Pearson

Pearson score:
0.6944217062199275
(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Adam Cohen" --score-type Euclidean

Euclidean score:
0.38742588672279304
(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Adam Cohen" --score-type Pearson

Pearson score:
0.9081082718950217
(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Chris Duncan" --score-type Euclidean

Euclidean score:
0.38742588672279304
(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> python LR_4_task_9.py --user1 "David Smith"
--user2 "Chris Duncan" --score-type Pearson

Pearson score:
1.0
```

Рис 4.29 Обрахунок оцінок

Завдання 2.10. Пошук користувачів зі схожими уподобаннями методом ко-лаборативної 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
```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

```

!= 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 = '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))

```

Результат виконання завдання:

```

(venv) PS C:\Users\dubin\OneDrive\Робочий стол\4 course\AI\lab4> 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

```

Рис 4.30 Знаходження найбільших оцінок

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
from LR_4_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)

```

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    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]

    return recommended_movies[:10]

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

```

Результат виконання завдання:

```

(venv) PS C:\Users\dubin\OneDrive\Рабочий стол\4 course\AI\lab4> python LR_4_task_11.py --user "Chris Duncan"
Movies recommended for Chris Duncan:
1. Vertigo
2. Scarface
3. Goodfellas
4. Roman Holiday

```

Рис 4.31 Знаходження найбільших оцінок

Посилання на GitHub: https://github.com/BogdanStelmah/Basics-of-AI_labs

Висновок: На даній лабораторній роботі ми використовуємо спеціалізовані бібліотеки та мову програмування Python дослідили методи ансамблів у машинному навчанні та створили рекомендаційні системи.

		Дубинченко Б.М.			ДУ «Житомирська політехніка».23.121.05.000 – Лр4	Арк.
		Іванов Д.А.				23
Змн.	Арк.	№ докум.	Підпис	Дата		