

## Лабораторна робота №6

### СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

**Мета:** використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися створювати рекомендаційні системи.

#### Хід роботи

Посилання на GitHub: [ross3005/ai\\_labs\\_hordeiev\\_pi60 \(github.com\)](https://github.com/ross3005/ai_labs_hordeiev_pi60)

#### Завдання №1:

Напишемо код для навчального конвеєра:

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

# Generate data
X, y = make_blobs(n_samples=150,
                  n_features=25, random_state=7)

# Select top K features
k_best_selector = SelectKBest(f_regression, k=9)

# Initialize Extremely Random Forests classifier
classifier = ExtraTreesClassifier(n_estimators=60, max_depth=4)

# Construct the pipeline
processor_pipeline = Pipeline([('selector', k_best_selector), ('erf',
                                                                classifier)])

# Set the parameters
processor_pipeline.set_params(selector__k=7, erf__n_estimators=30)

# Training the pipeline
processor_pipeline.fit(X, y)

# Predict outputs for the input data
output = processor_pipeline.predict(X)
print("\nPredicted output:\n", output)

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

# Print the features chosen by the pipeline selector
status = processor_pipeline.named_steps['selector'].get_support()

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

Результат:

					Житомирська політехніка.22.121.4.000 – Лр6			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Розроб.		Гордєєв Р.С.						
Перевір.		Філіпов В.О.					1	16
Керівник						ФІКТ Гр. ПІ-60[1]		
Н. контр.								
Зав. каф.								

```

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

Score: 1.0

Indices of selected features: 5, 7, 10, 15, 16, 17, 21

```

Рис. 1. Результат навчального конвеєра

Перше отримане значення представляє собою результуючі мітки. Значення Score характеризує ефективність процесора даних. В останній стрічці відображені індекси відібраних властивостей.

## Завдання №2:

Напишемо код для пошуку найближчих сусідів:

```

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

# Input data
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]])

# Number of nearest neighbors
k = 5

# Test datapoint
test_datapoint = [4.3, 2.7]

# Plot input data
plt.figure()
plt.title('Input data')
plt.scatter(X[:,0], X[:,1], marker='o', s=75, color='black')

# Build K Nearest Neighbors model
knn_model = NearestNeighbors(n_neighbors=k, algorithm='ball_tree').fit(X)
distances, indices = knn_model.kneighbors([test_datapoint])

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

# Visualize the nearest neighbors along with the test datapoint
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')

```

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```
plt.show()
```

Результати:

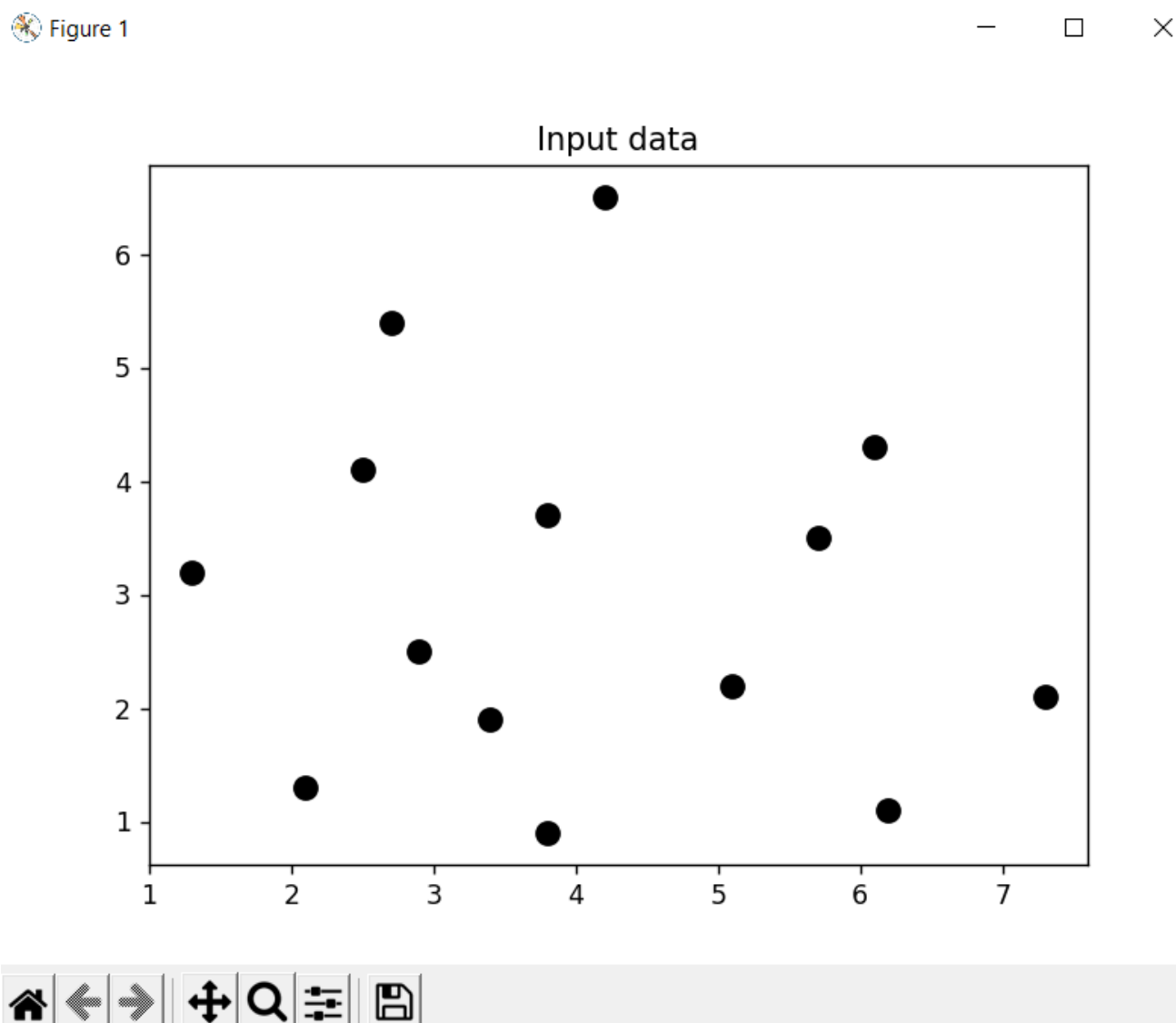


Рис. 2. Графік вхідних даних

		Гордеев Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

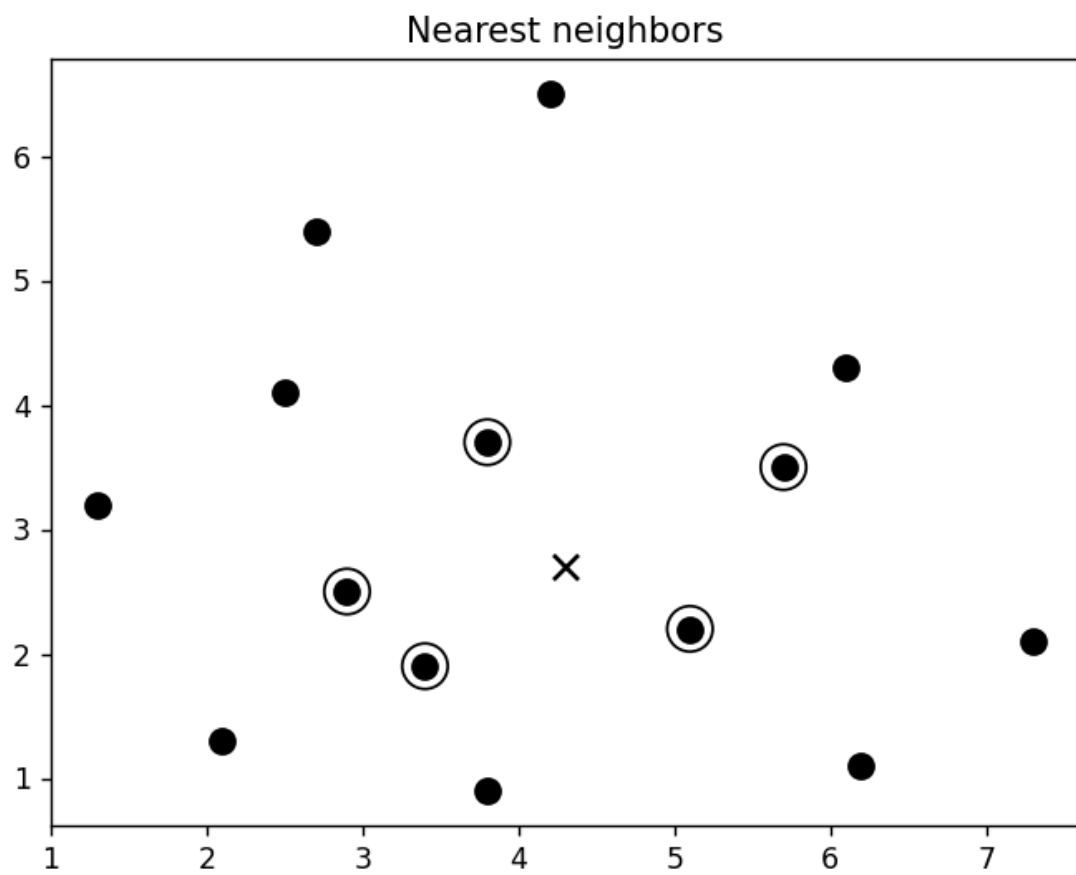


Рис. 3. Графік найближчих сусідів

```
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. Результат пошуку найближчих сусідів

На першому графіку відображено вхідні дані. На другому графіку позначено п'ять найближчих сусідів. Тестова точка позначена хрестиком, а найближчі точки обведені колом. В вікні терміналу показується результат пошуку найближчих сусідів.

### Завдання №3:

Напишемо код для створення класифікатора методом k найближчих сусідів:

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```

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

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

# Plot input data
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')

# Number of nearest neighbors
num_neighbors = 12

# Step size of the visualization grid
step_size = 0.01

# Create a K Nearest Neighbours classifier model
classifier = neighbors.KNeighborsClassifier(num_neighbors, weights='distance')

# Train the K Nearest Neighbours model
classifier.fit(X, y)

# Create the mesh to plot the boundaries
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))

# Evaluate the classifier on all the points on the grid
output = classifier.predict(np.c_[x_values.ravel(), y_values.ravel()])

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

# Overlay the training points on the map
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=50, edgecolors='black', facecolors='none')

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

# Test input datapoint
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',

```

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        linewidth=6, s=200, facecolors='black')

# Extract the K nearest neighbors
_, indices = classifier.kneighbors([test_datapoint])
indices = indices.astype(np.int)[0]

# Plot k nearest neighbors
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()

```

Результати:

Figure 1

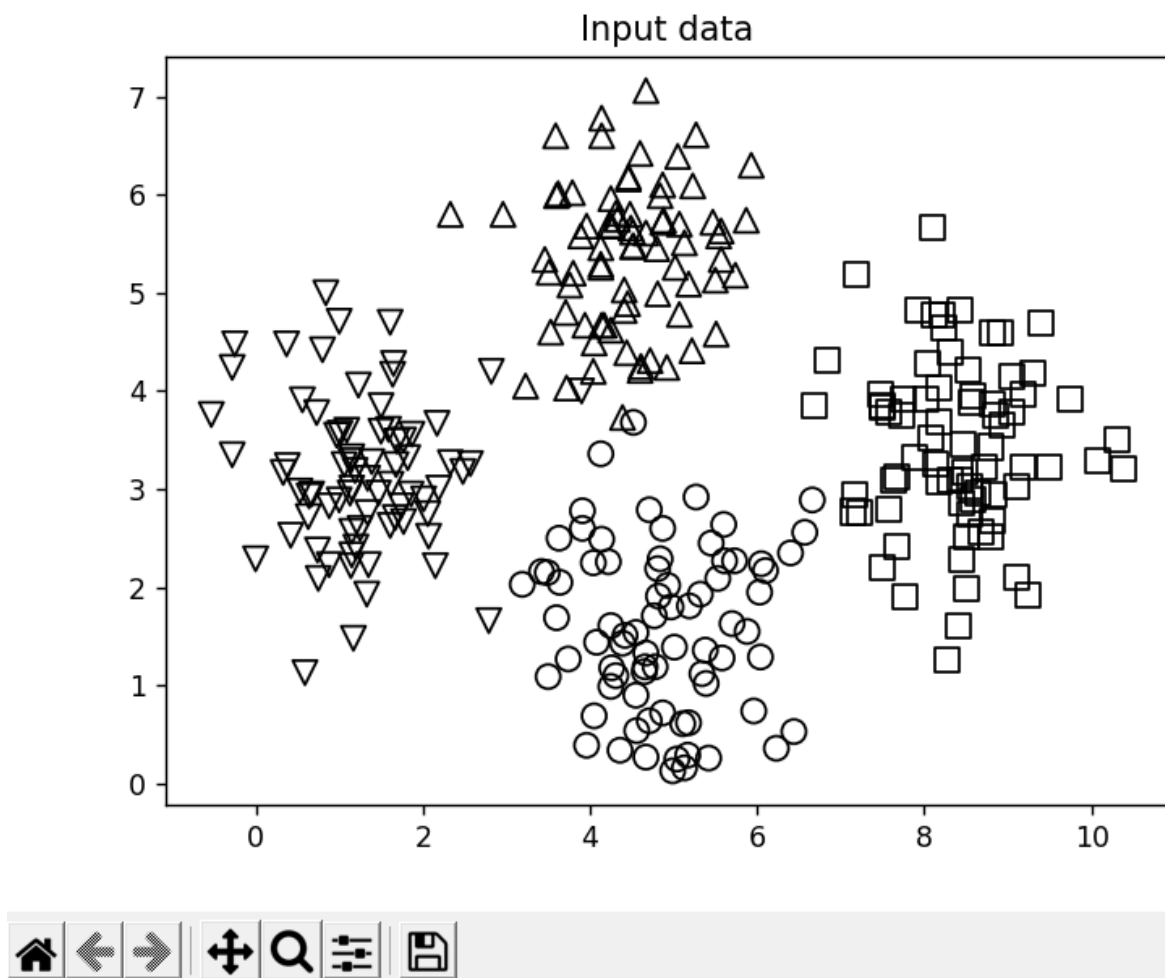


Рис. 5. Графік вхідних даних

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

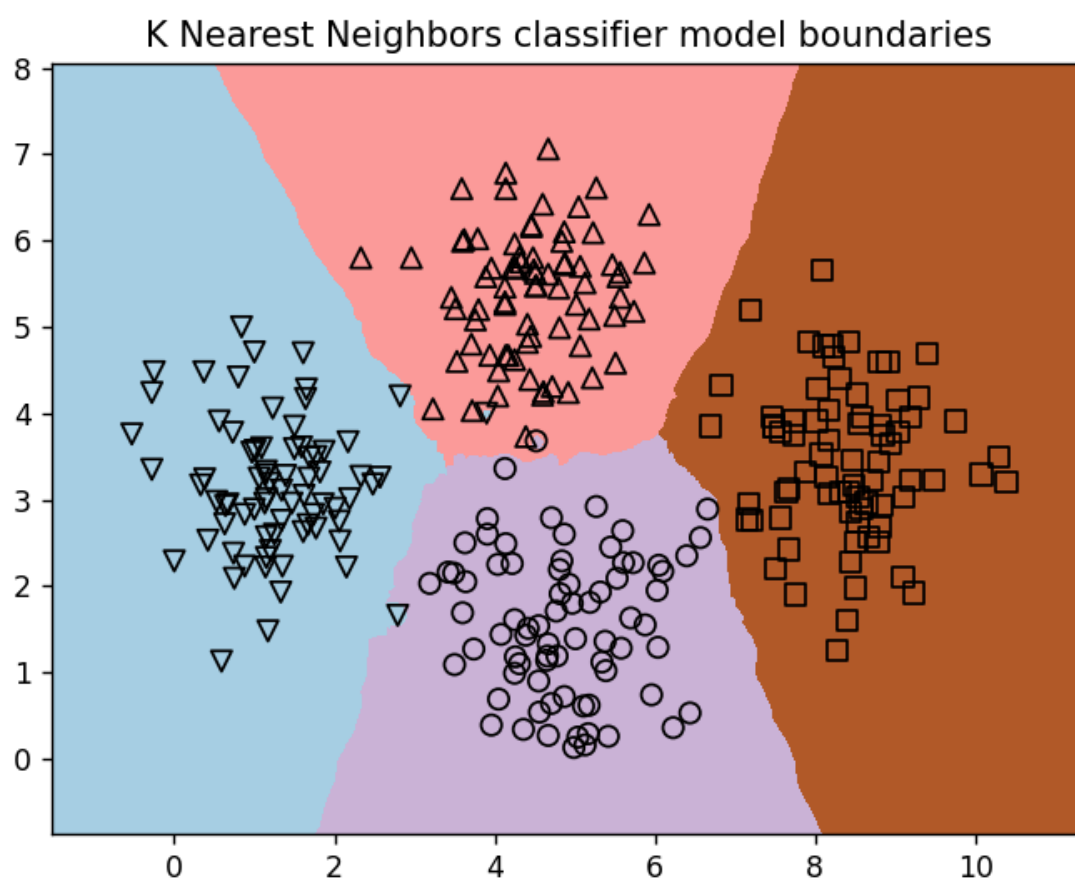


Рис. 6. Графік кордонів класифікатора

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

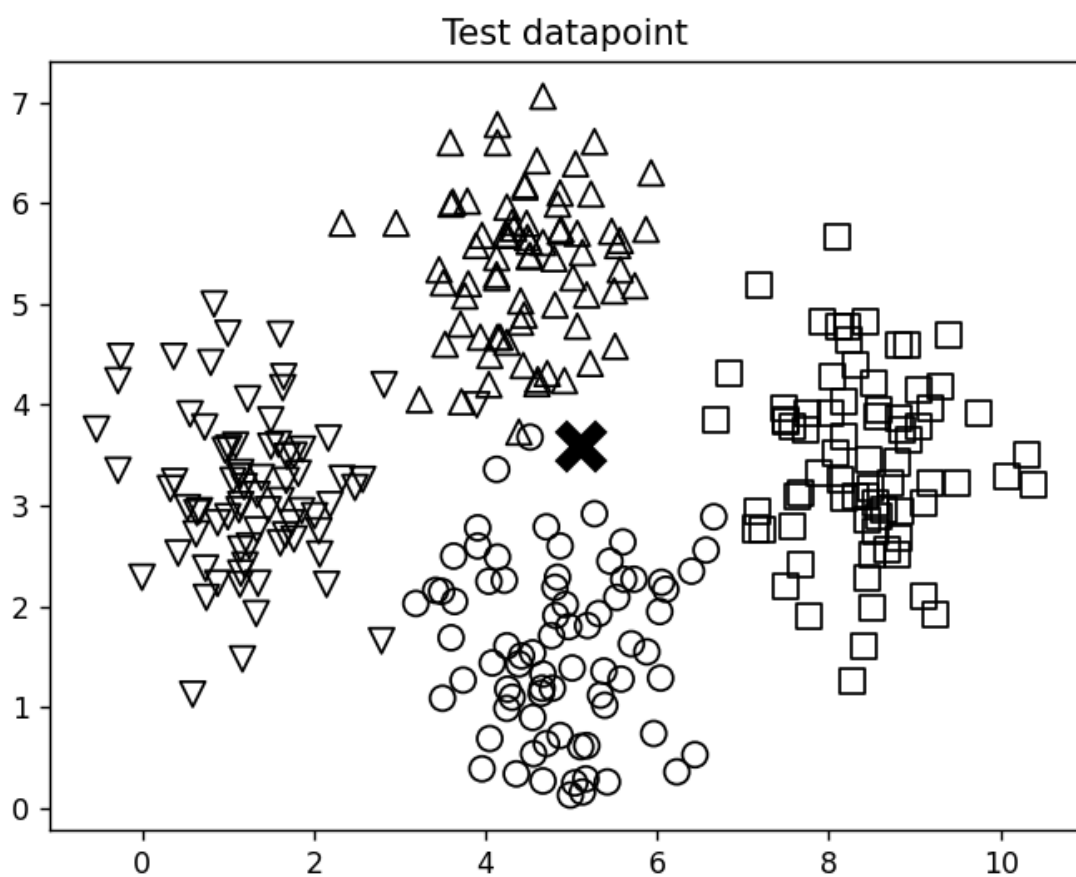


Рис. 7. Графік положення тестової точки даних (позначена хрестиком)



Figure 4

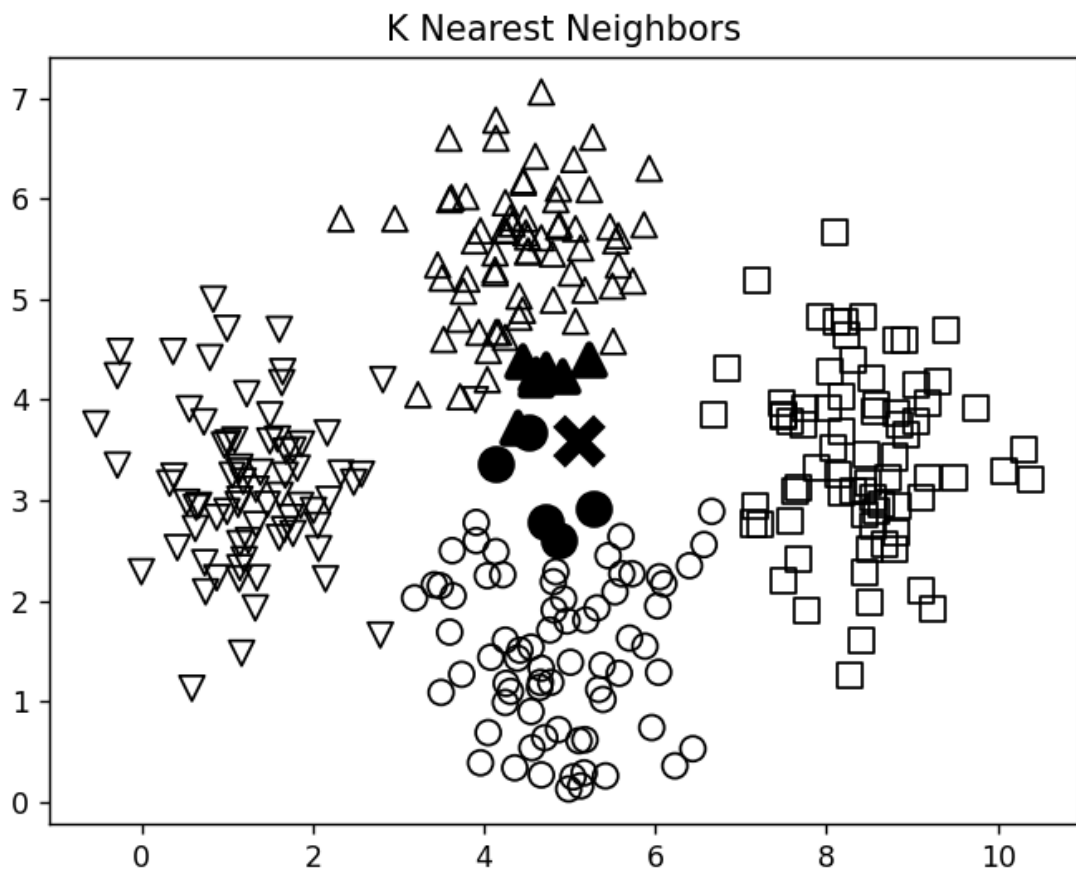


Рис. 8. Графік з 12 найближчими сусідами тестової точки

Predicted output: 1

Рис. 9. Спрогнозований результат

Отже, тестова точка відноситься до класу 1.

#### Завдання №4:

Напишемо код для 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
```

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

```

# Compute the Euclidean distance score between user1 and 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')

    # Movies rated by both user1 and user2
    common_movies = {}

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

    # If there are no common movies between the users,
    # then the score is 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)))

# Compute the Pearson correlation score between user1 and user2
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')

    # Movies rated by both user1 and user2
    common_movies = {}

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

    num_ratings = len(common_movies)

    # If there are no common movies between user1 and user2, then the score is 0
    if num_ratings == 0:
        return 0

    # Calculate the sum of ratings of all the common movies
    user1_sum = np.sum([dataset[user1][item] for item in common_movies])
    user2_sum = np.sum([dataset[user2][item] for item in common_movies])

    # Calculate the sum of squares of ratings of all the 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])

```

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Calculate the sum of products of the ratings of the common movies
sum_of_products = np.sum([dataset[user1][item] * dataset[user2][item] for item
in common_movies])

# Calculate the Pearson correlation score
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))

```

Результати:

```

Euclidean score:
0.585786437626905

```

Рис. 10. Результат обчислення евклідової оцінки подібності користувачів David Smith та Bill Duffy

```

Pearson score:
0.9909924304103233

```

Рис. 11. Результат обчислення оцінки подібності Пірсона для користувачів David Smith та Bill Duffy

```

Euclidean score:
0.1424339656566283
PS C:\Users\win34\Py

Pearson score:
-0.7236759610155113

```

Рис. 12. Результат обчислення евклідової оцінки та оцінки подібності Пірсона для користувачів David Smith та Brenda Peterson

		Гордеев Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Euclidean score:
0.30383243470068705
PS C:\Users\win34\Py

Pearson score:
0.7587869106393281
```

Рис. 13. Результат обчислення евклідової оцінки та оцінки подібності Пірсона для користувачів David Smith та Samuel Miller

```
Euclidean score:
0.2857142857142857
PS C:\Users\win34\Py

Pearson score:
0
```

Рис. 14. Результат обчислення евклідової оцінки та оцінки подібності Пірсона для користувачів David Smith та Julie Hammel

```
Euclidean score:
0.28989794855663564
PS C:\Users\win34\Py

Pearson score:
0.6944217062199275
```

Рис. 15. Результат обчислення евклідової оцінки та оцінки подібності Пірсона для користувачів David Smith та Clarissa Jackson

```
Euclidean score:
0.38742588672279304
PS C:\Users\win34\Py

Pearson score:
0.9081082718950217
```

Рис. 16. Результат обчислення евклідової оцінки та оцінки подібності Пірсона для користувачів David Smith та Adam Cohen

		Гордеев Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Euclidean score:
0.38742588672279304
PS C:\Users\win34\Py

Pearson score:
1.0
```

Рис. 17. Результат обчислення евклідової оцінки та оцінки подібності Пірсона для користувачів David Smith та Chris Duncan

Отже, судячи з графіку, найважливішим параметром є LSTAT.

### Завдання №5:

Напишемо код для пошуку користувачів зі схожими уподобаннями методом колаборативної фільтрації:

```
import argparse
import json
import numpy as np

from compute_scores 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

# Finds users in the dataset that are similar to the input user
def find_similar_users(dataset, user, num_users):
    if user not in dataset:
        raise TypeError('Cannot find ' + user + ' in the dataset')

    # Compute Pearson score between input user
    # and all the users in the dataset
    scores = np.array([[x, pearson_score(dataset, user, x)] for x in dataset if x != user])

    # Sort the scores in decreasing order
    scores_sorted = np.argsort(scores[:, 1])[::-1]

    # Extract the top 'num_users' scores
    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)
```

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```
print('User\t\t\tSimilarity score')
print('-'*41)
for item in similar_users:
    print(item[0], '\t\t\t', round(float(item[1]), 2))
```

Результати:

```
Users similar to Bill Duffy:

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

Рис. 18. Користувачі аналогічні користувачеві Bill Duffy

```
Users similar to Clarissa Jackson:

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

Рис. 19. Користувачі аналогічні користувачеві Clarissa Jackson

### Завдання №6:

Напишемо код для створення рекомендаційної системи фільмів:

```
import argparse
import json
import numpy as np

from compute_scores import pearson_score
from collaborative_filtering import find_similar_users

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

# Get movie recommendations for the input user
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']

# Generate movie ranks by normalization
movie_scores = np.array([[score / similarity_scores[item], item]
                          for item, score in overall_scores.items()])

# Sort in decreasing order
movie_scores = movie_scores[np.argsort(movie_scores[:, 0])[::-1]]

# Extract the movie recommendations
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)

```

Результати:

```

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

```

Рис. 20. Рекомендаційні фільми для користувача Chris Duncan

```

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

```

Рис. 21. Рекомендаційні фільми для користувача Julie Hammel

		Гордєєв Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

**Висновок:** на цій лабораторній роботі я, використовуючи спеціалізовані бібліотеки та мову програмування Python, навчився створювати рекомендаційні системи.

		Гордеев Р.С.			Житомирська політехніка.22.121.4.000 – Лр6	Арк.
		Філіпов В.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		