ЗВІТ

З лабораторної роботи №4

ДОСЛІДЖЕННЯ МЕТОДІВ АНСАМБЛЕВОГО НАВЧАННЯ ТА СТВОРЕННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

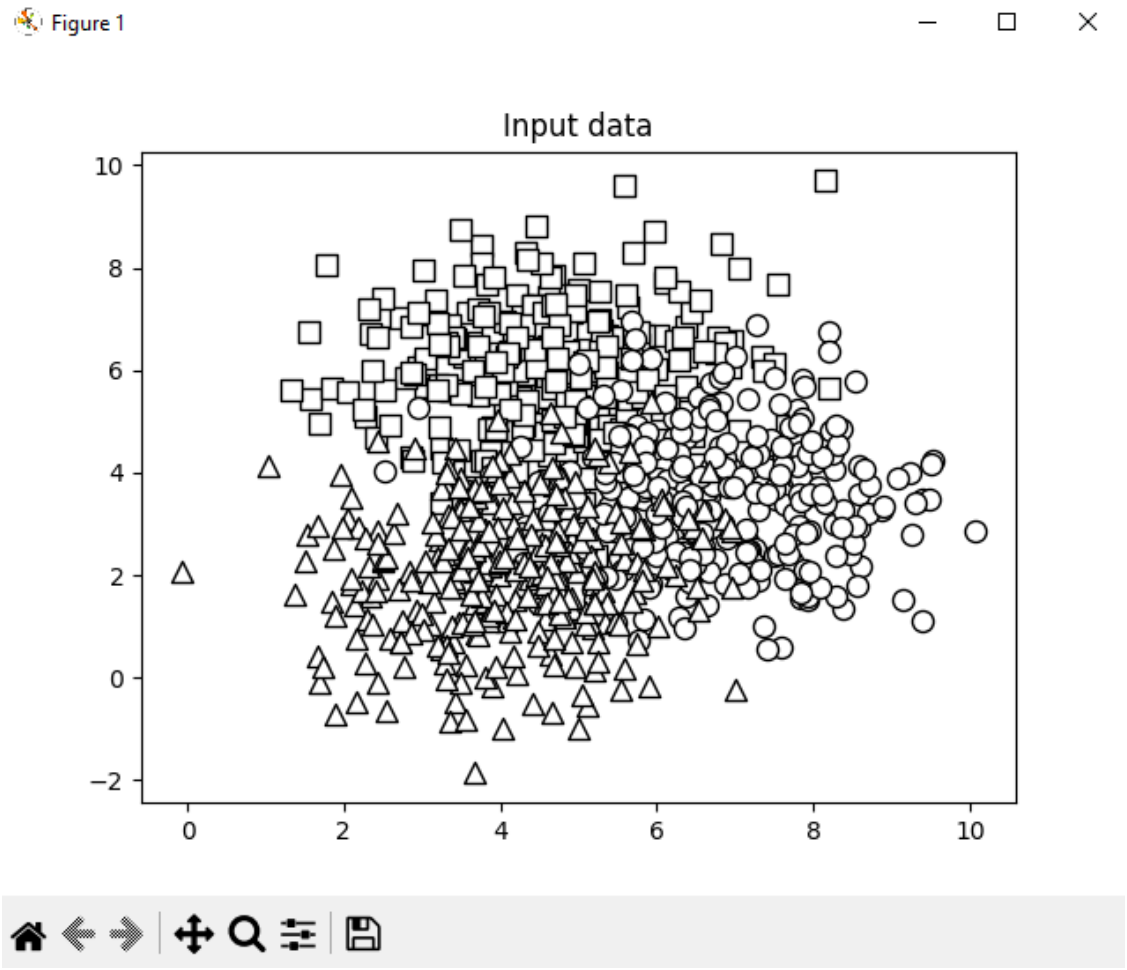
Студента КН-20-1 навчальної групи

Кірія Даніли Олеговича варіант №6

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

Завдання 1.

Classifier-type “rf”



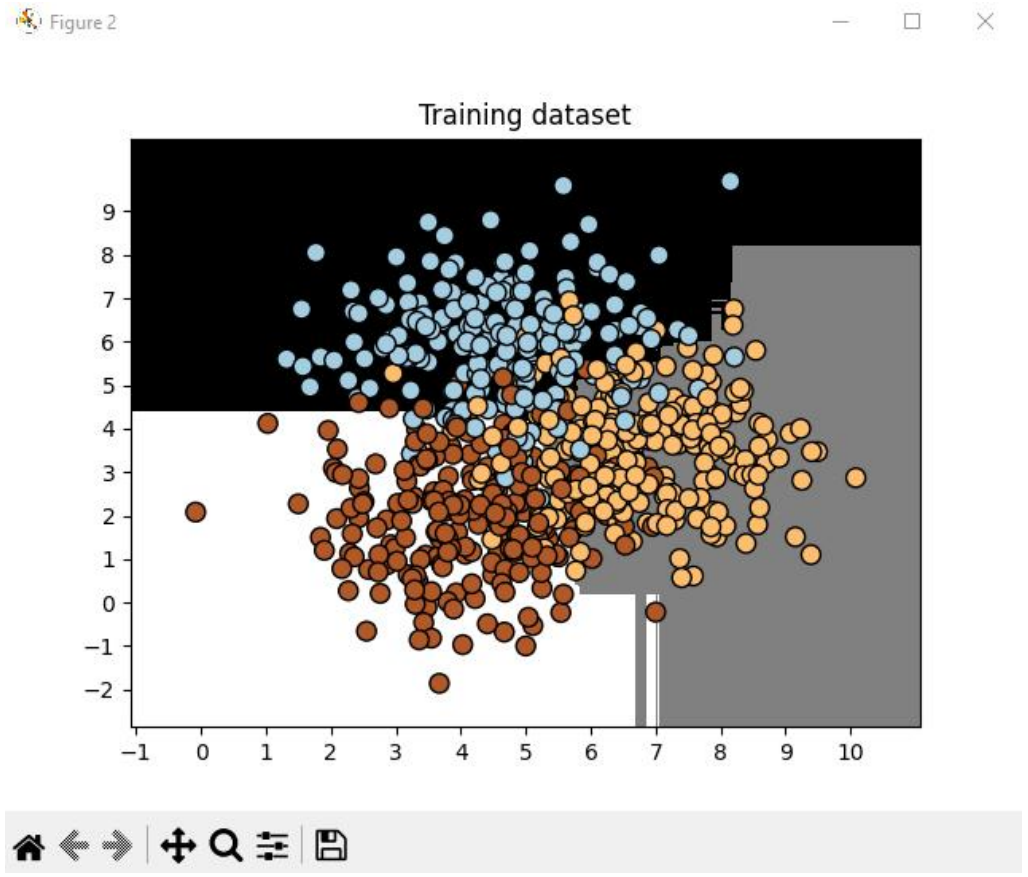


Figure 2
Training dataset
9
8
7
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6
7
8
9
10

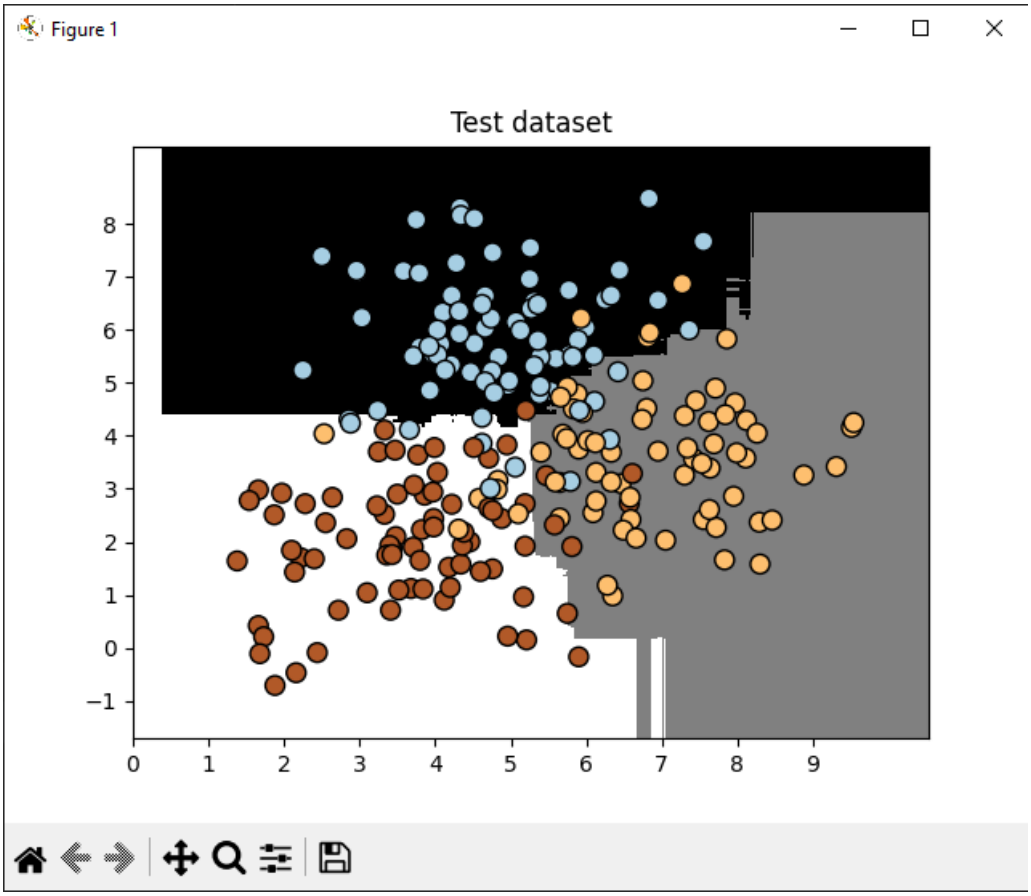
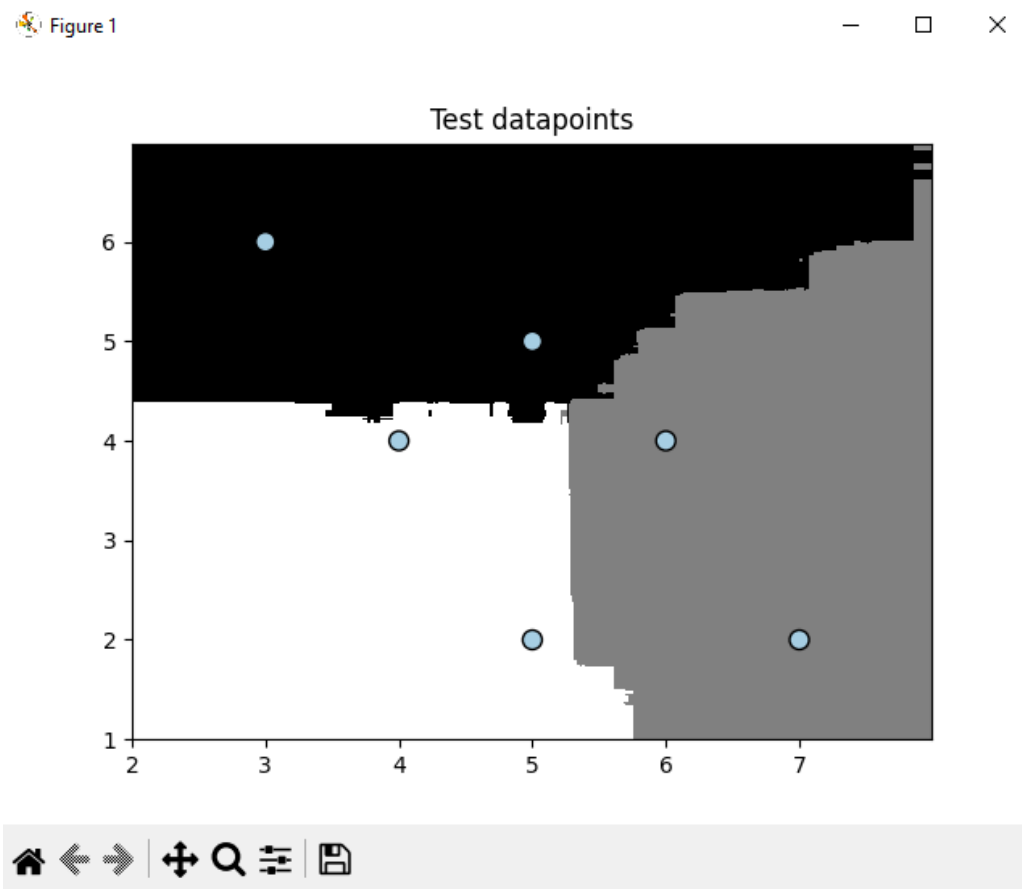


Figure 1
Test dataset
8
7
6
5
4
3
2
1
0
−1
0
1
2
3
4
5
6
7
8
9




```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_1.py --classifier-type "rf"

########################################

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

########################################

########################################

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
```

Classifier-type “erf”

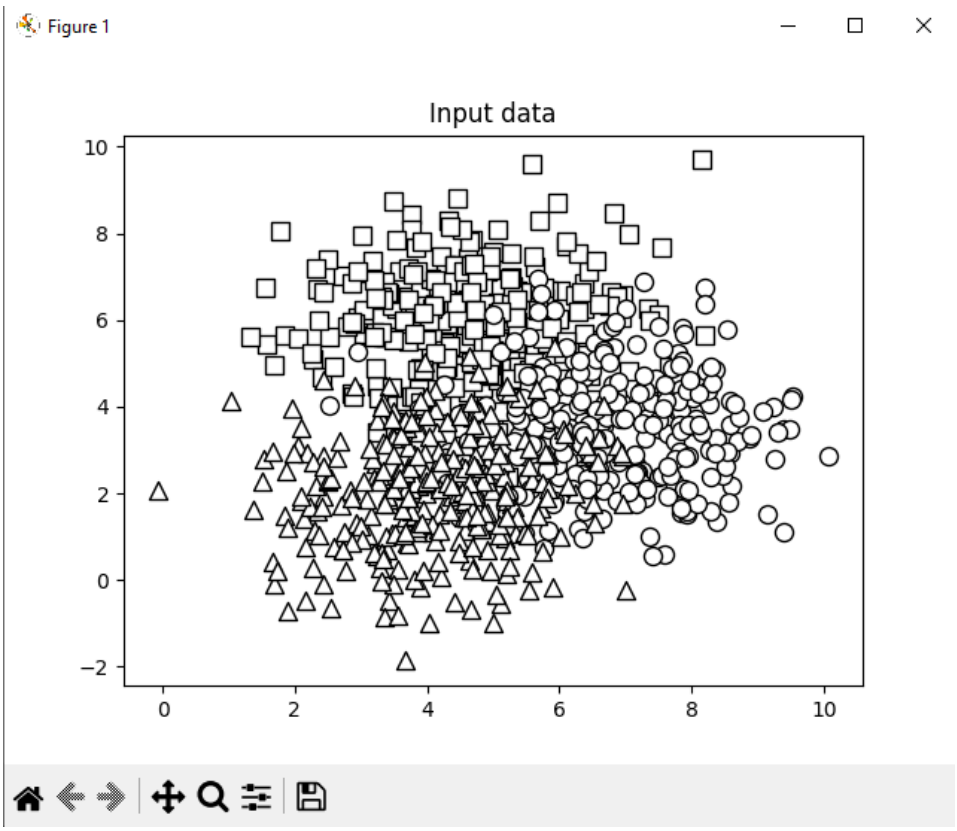


Figure 1
Input data
10
8
6
4
2
0
−2
0
2
4
6
8
10

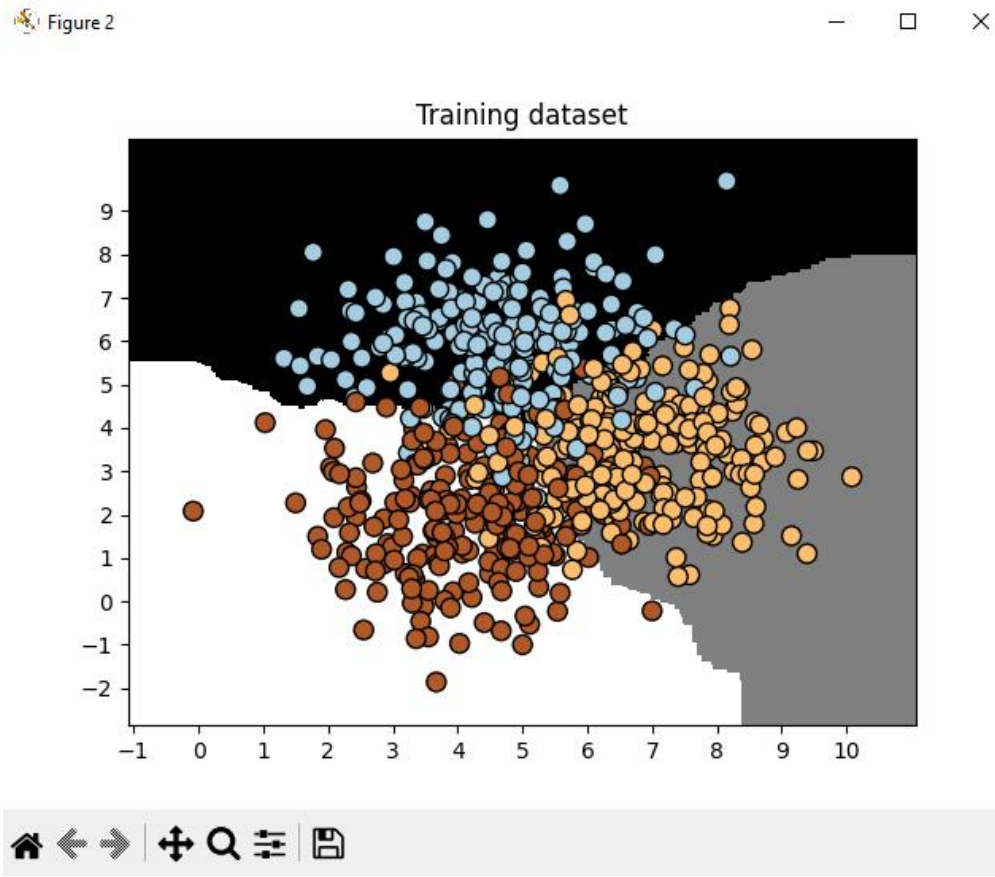


Figure 2
Training dataset
9
8
7
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6
7
8
9
10

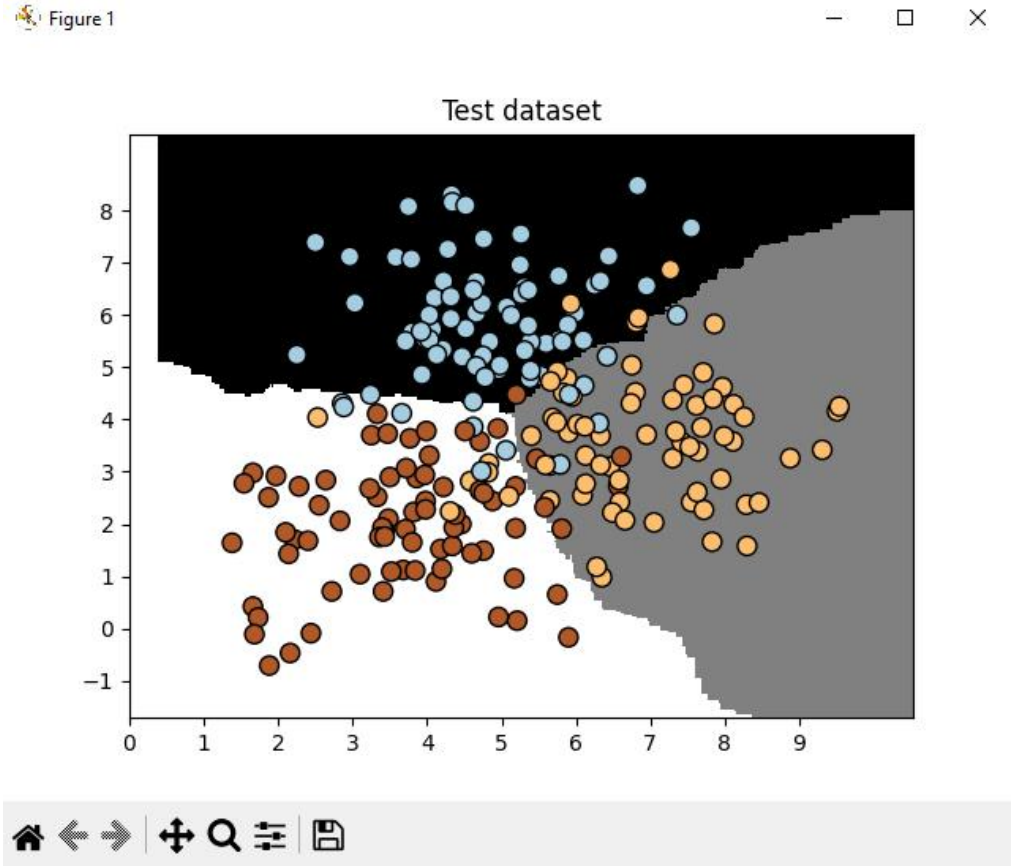


Figure 1
Test dataset
8
7
6
5
4
3
2
1
0
−1
0
1
2
3
4
5
6
7
8
9

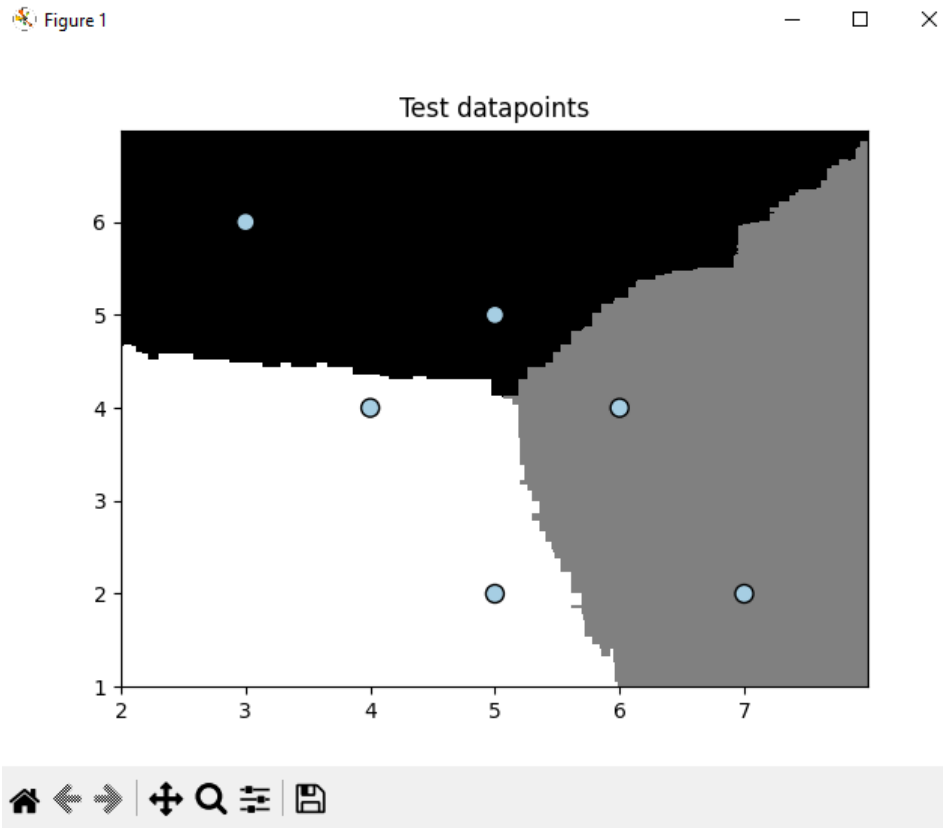


Figure 1
Test datapoints
6
5
4
3
2
1
2
3
4
5
6
7

```
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

########################################


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
```

Код програми:

```
import argparse

import matplotlib
import numpy as np
import matplotlib.pyplot as plt
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

def build_arg_parser():
    parser = argparse.ArgumentParser(description='Classify data using Ensemble
Learning techniques')
    parser.add_argument('--classifier-type', dest='classifier_type',
required=True, choices=['rf', 'erf'],
                        help="Type of classifier to use; can be either 'rf' or
'erf'")
    return parser


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

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

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

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

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

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

    test_datapoints = np.array([[5, 5], [3, 6], [6, 4], [7, 2], [4, 4], [5, 2]])

    print("\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)

    visualize_classifier(classifier, test_datapoints, [0] *
len(test_datapoints), 'Test datapoints')
    plt.show()
```

Завдання 2:

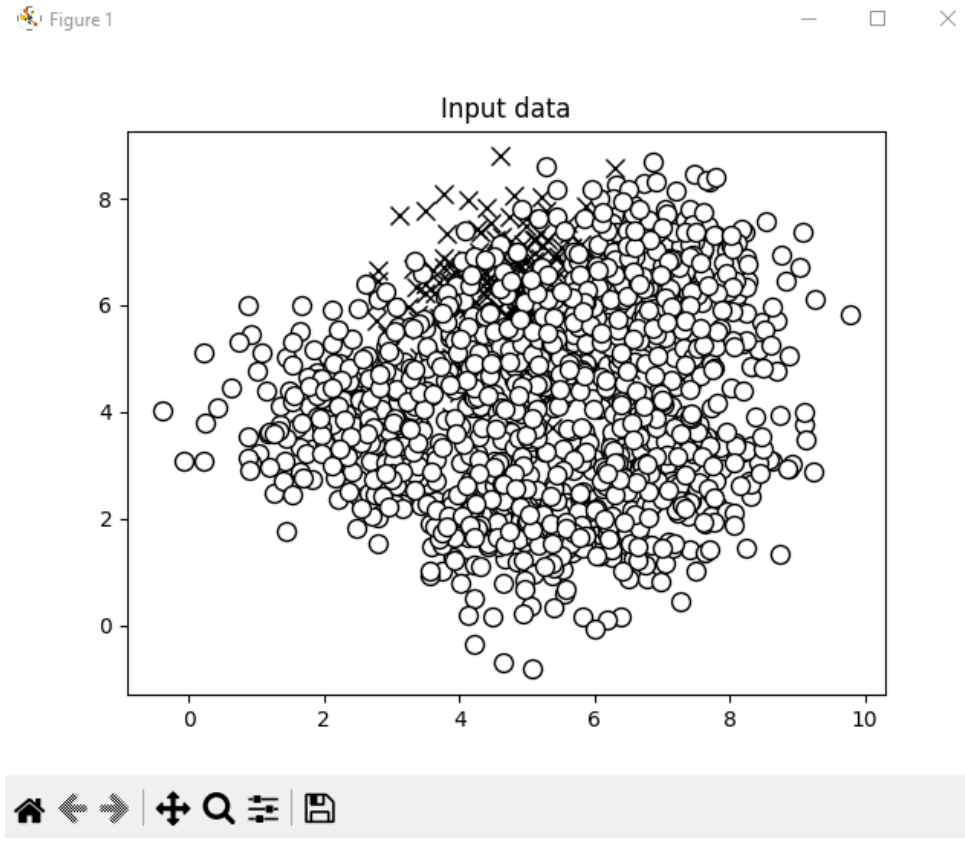


Figure 1
Input data
0
2
4
6
8
10

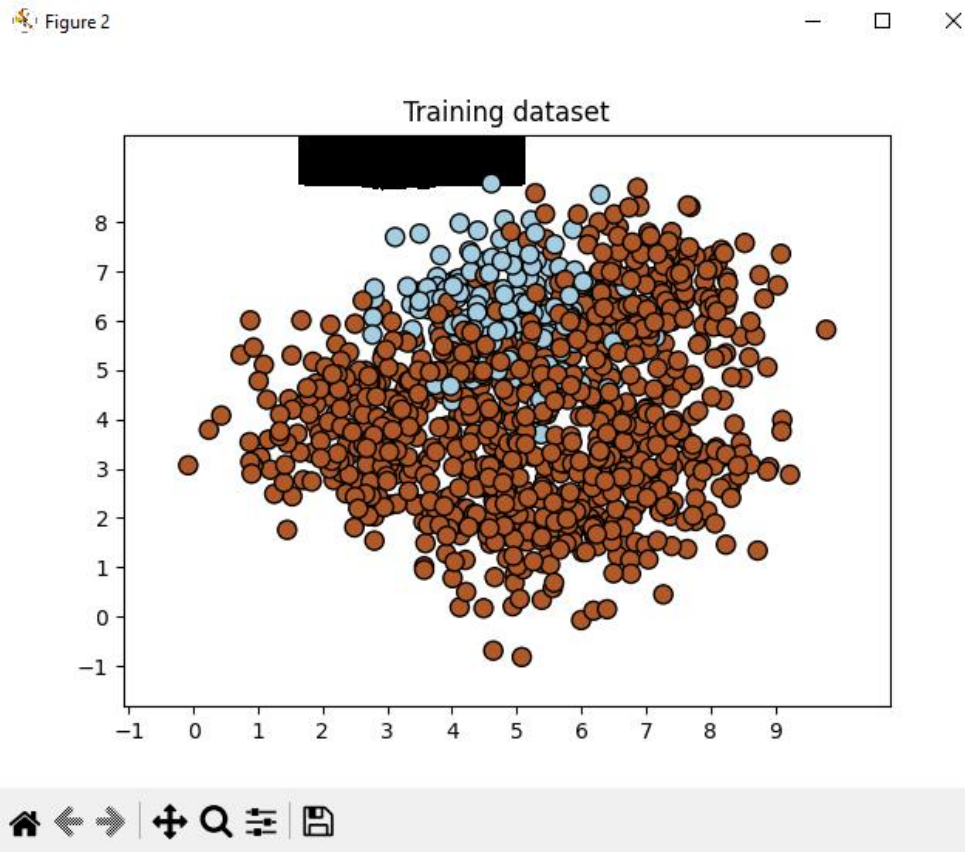
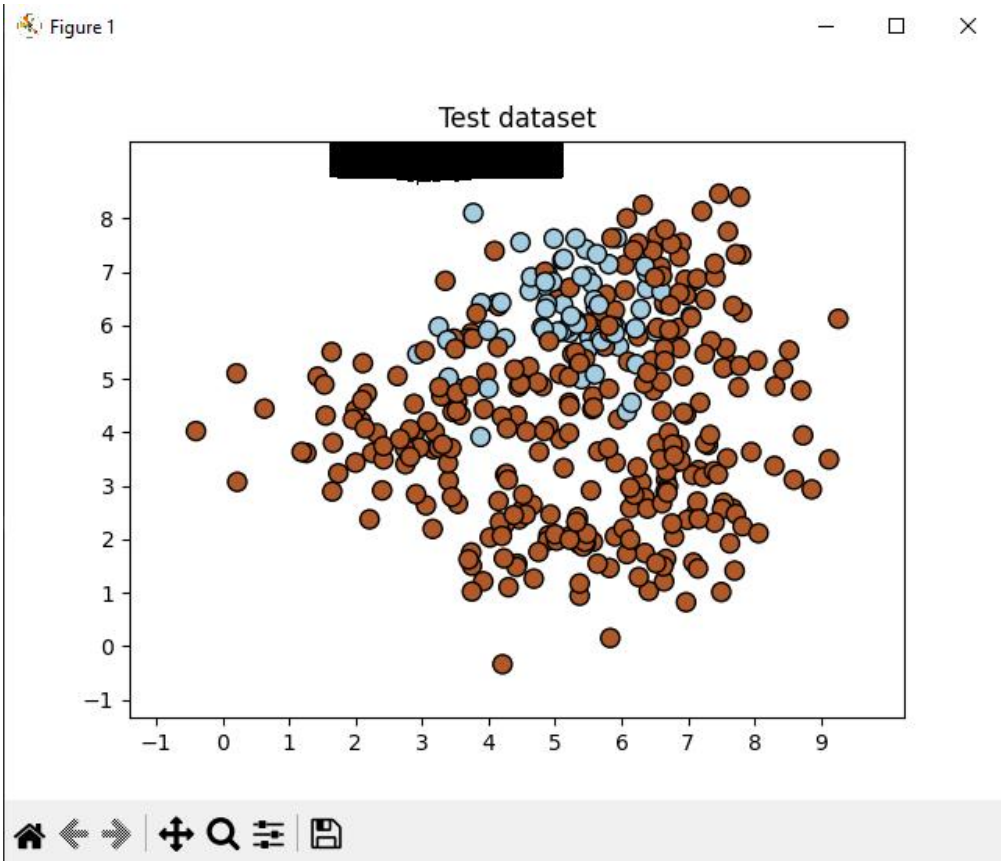


Figure 2
Training dataset
−1
0
1
2
3
4
5
6
7
8
9




```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py -m LR_4_task_2.py
D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4\LR_4_task_2.py:17: UserWarning: You passed a ed
s behavior may change in the future.
  plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='black',

########################################

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       1.00      0.01      0.01       181
     Class-1       0.84      1.00      0.91       944

    accuracy                           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
```

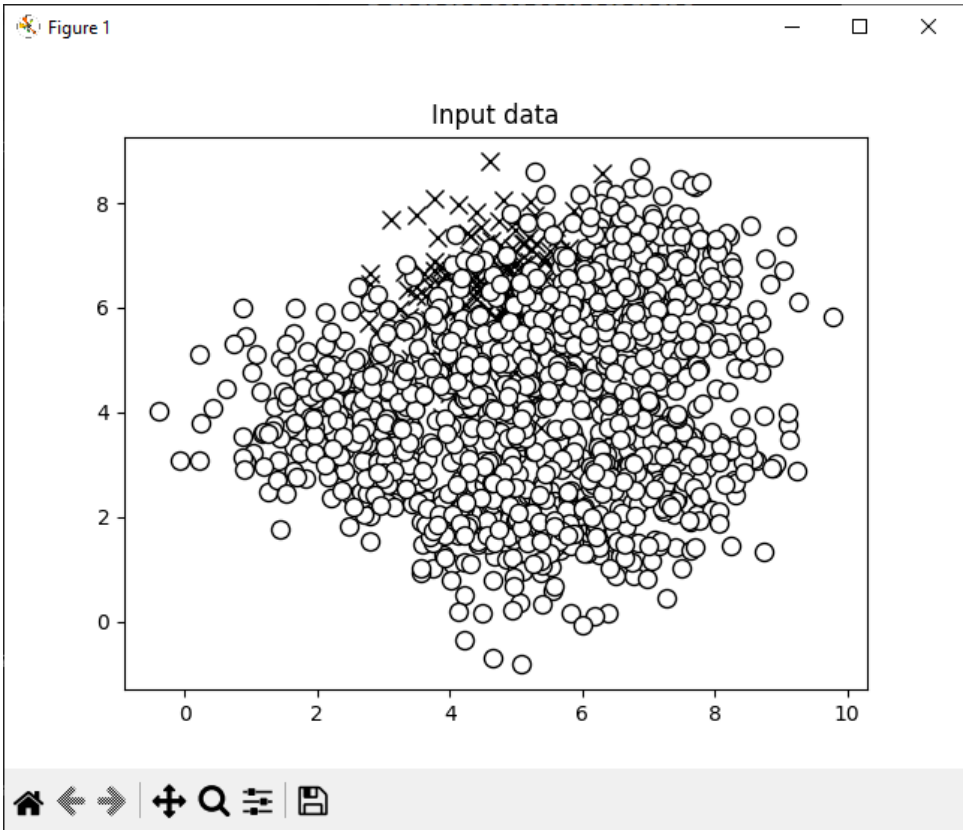


Figure 1
Input data
0
2
4
6
8
10

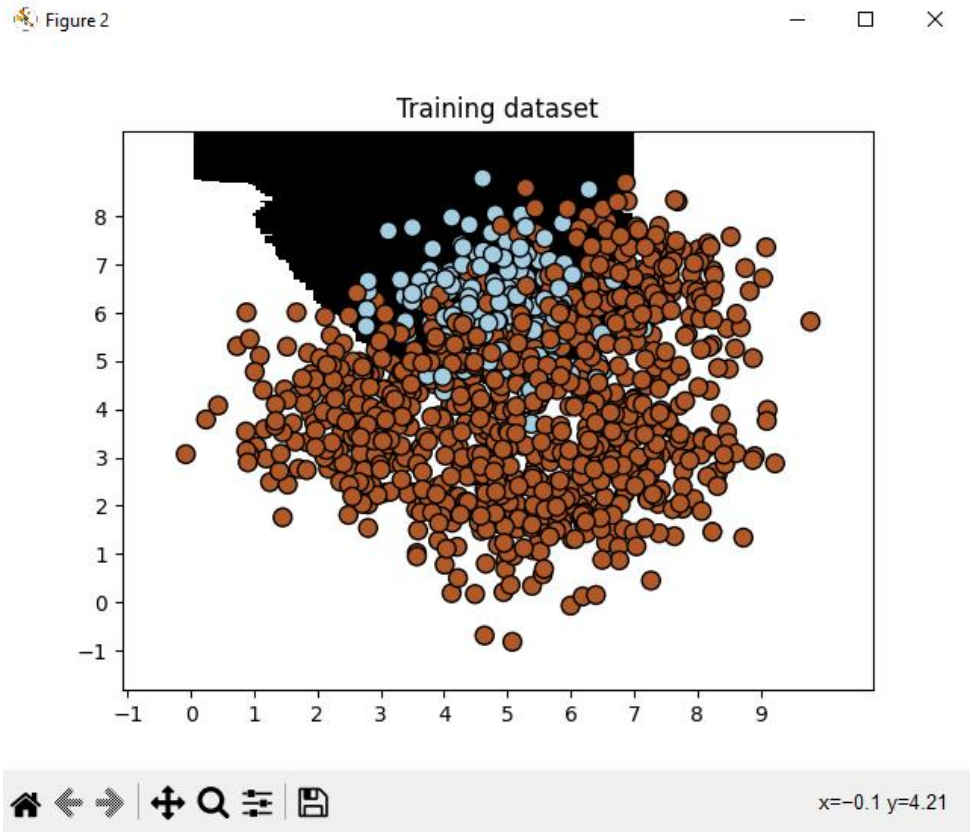
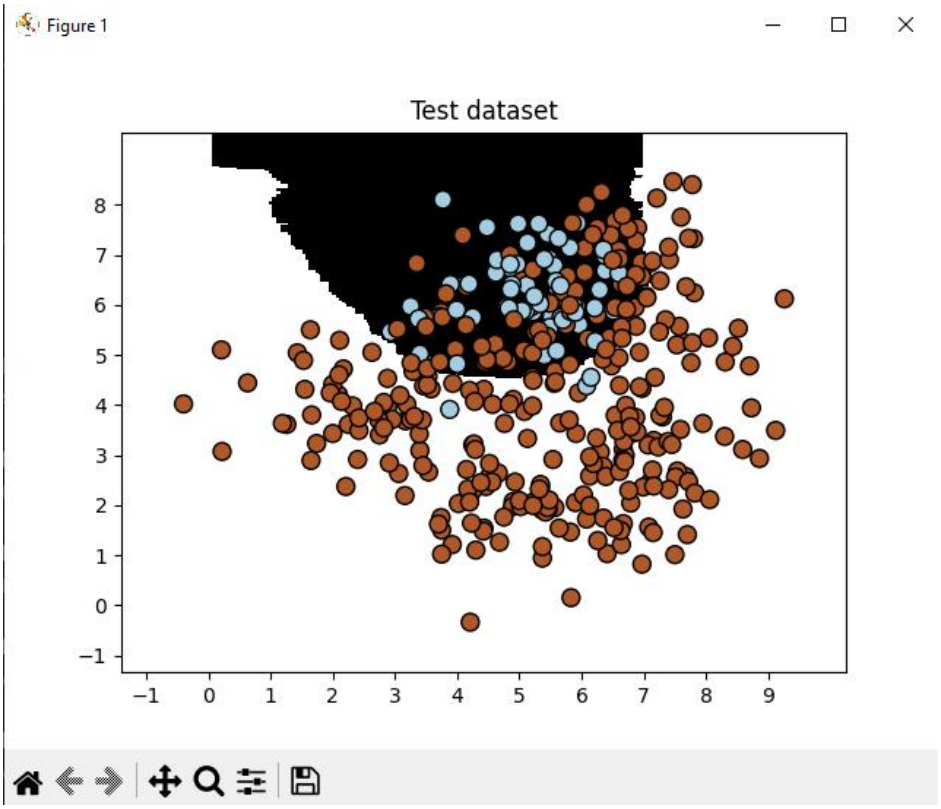


Figure 2
Training dataset
−1
0
1
2
3
4
5
6
7
8
9
x=−0.1 y=4.21




```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py -m LR_4_task_2.py balance
D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4\LR_4_task_2.py:17: UserWarning:
s behavior may change in the future.
  plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='black',

########################################

Classifier performance on training dataset

              precision    recall  f1-score   support

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

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

########################################

########################################

Classifier performance on test dataset

              precision    recall  f1-score   support

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

    accuracy                           0.78       375
   macro avg       0.72      0.84      0.73       375
```

Код програми:

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

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

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

plt.figure()
plt.scatter(class_0[:, 0], class_0[:, 1], s=75, 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 = {'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')

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()
```

Завдання 3:

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_3.py

##### Searching optimal parameters for precision_weighted
   mean_score_time  std_score_time param_max_depth param_n_estimators                                params  ...  split3_test_score  split4_test_score  mean_test_score  std_test_score  rank_test_score
0         0.009200    4.000187e-04               2                100   {'max_depth': 2, 'n_estimators': 100}  ...           0.820784           0.856884         0.849757        0.025132                1
1         0.009400    4.899015e-04               4                100   {'max_depth': 4, 'n_estimators': 100}  ...           0.802601           0.854124         0.841135        0.023032                5
2         0.009400    8.002758e-04               7                100   {'max_depth': 7, 'n_estimators': 100}  ...           0.801926           0.860624         0.843820        0.026560                4
3         0.011600    2.244602e-03              12                100  {'max_depth': 12, 'n_estimators': 100}  ...           0.803514           0.853896         0.831955        0.028334                8
4         0.011400    1.020076e-03              16                100  {'max_depth': 16, 'n_estimators': 100}  ...           0.794215           0.860232         0.816484        0.033429                9
5         0.004000    1.168008e-07               4                 25    {'max_depth': 4, 'n_estimators': 25}  ...           0.793105           0.868577         0.845574        0.029698                2
6         0.005600    7.998112e-04               4                 50    {'max_depth': 4, 'n_estimators': 50}  ...           0.802601           0.863329         0.839851        0.020401                7
7         0.009200    3.999236e-04               4                100   {'max_depth': 4, 'n_estimators': 100}  ...           0.802601           0.854124         0.841135        0.023032                5
8         0.021399    2.153990e-03               4                250   {'max_depth': 4, 'n_estimators': 250}  ...           0.801926           0.854124         0.844788        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       225
   macro avg       0.86      0.86      0.86       225
weighted avg       0.86      0.86      0.86       225
```

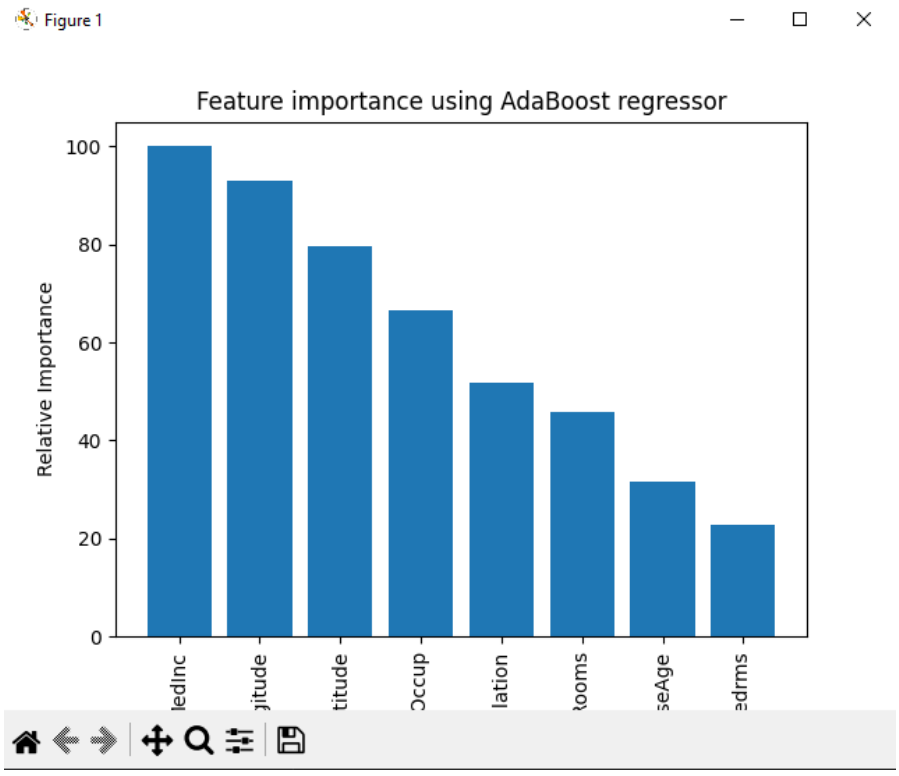
```
##### Searching optimal parameters for recall_weighted
   mean_score_time  std_score_time param_max_depth param_n_estimators                                params  ...  split3_test_score  split4_test_score  mean_test_score  std_test_score  rank_test_score
0         0.009200        0.000749               2                100   {'max_depth': 2, 'n_estimators': 100}  ...           0.814815           0.851852         0.842963        0.027075                1
1         0.019399        0.019334               4                100   {'max_depth': 4, 'n_estimators': 100}  ...           0.800000           0.851852         0.837037        0.022468                5
2         0.017599        0.006343               7                100   {'max_depth': 7, 'n_estimators': 100}  ...           0.800000           0.859259         0.841481        0.026749                3
3         0.013799        0.001720              12                100  {'max_depth': 12, 'n_estimators': 100}  ...           0.800000           0.851852         0.829630        0.028497                8
4         0.012599        0.002870              16                100  {'max_depth': 16, 'n_estimators': 100}  ...           0.792593           0.859259         0.814815        0.034744                9
5         0.006000        0.002097               4                 25    {'max_depth': 4, 'n_estimators': 25}  ...           0.792593           0.866667         0.842963        0.029407                1
6         0.005599        0.000799               4                 50    {'max_depth': 4, 'n_estimators': 50}  ...           0.800000           0.859259         0.835556        0.020096                7
7         0.010000        0.002530               4                100   {'max_depth': 4, 'n_estimators': 100}  ...           0.800000           0.851852         0.837037        0.022468                5
8         0.021000        0.002098               4                250   {'max_depth': 4, 'n_estimators': 250}  ...           0.800000           0.851852         0.841481        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       225
   macro avg       0.86      0.86      0.86       225
weighted avg       0.86      0.86      0.86       225

PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

Код програми:

```python
import numpy as np
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
import pandas as pd

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

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

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

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

```
                ]

metrics = ['precision_weighted', 'recall_weighted']

for metric in metrics:
    print("\n##### Searching 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))
```

Завдання 4




```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_4.py

ADABOOST REGRESSOR
Mean squared error = 1.18
Explained variance score = 0.47
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.datasets import fetch_california_housing
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 = fetch_california_housing()

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

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

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

y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)

print("\nADABOOST REGRESSOR")
print(f"Mean squared error = {round(mse, 2)}")
print(f"Explained variance score = {round(evs, 2)}")

feature_importances = regressor.feature_importances_
feature_names = housing_data.feature_names

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

index_sorted = np.flipud(np.argsort(feature_importances))

pos = range(len(feature_names))

plt.figure()
plt.bar(pos, feature_importances[index_sorted], align='center')
plt.xticks(pos, [feature_names[i] for i in index_sorted], rotation=90)
plt.ylabel('Relative Importance')
plt.title('Feature importance using AdaBoost regressor')
plt.show()
```

Завдання 5:

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_5.py
Mean absolute error: 7.42
Predicted traffic: 26
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

Код програми:

```
import numpy as np
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

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

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

Завдання 6:

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_6.py

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

Score: 0.8466666666666667

Indices of selected features: 4, 7, 8, 12, 14, 17, 22
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

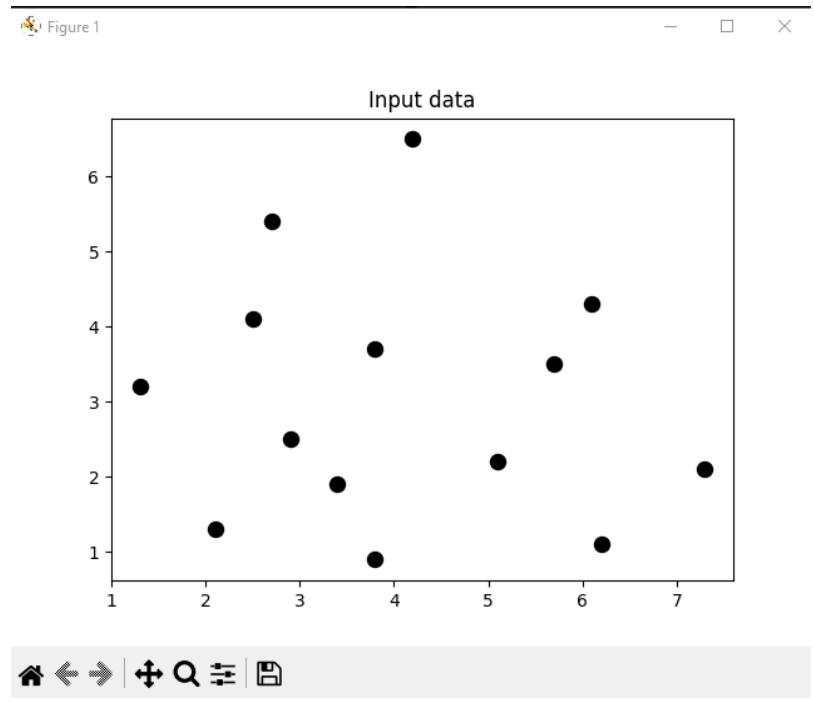
Код програми:

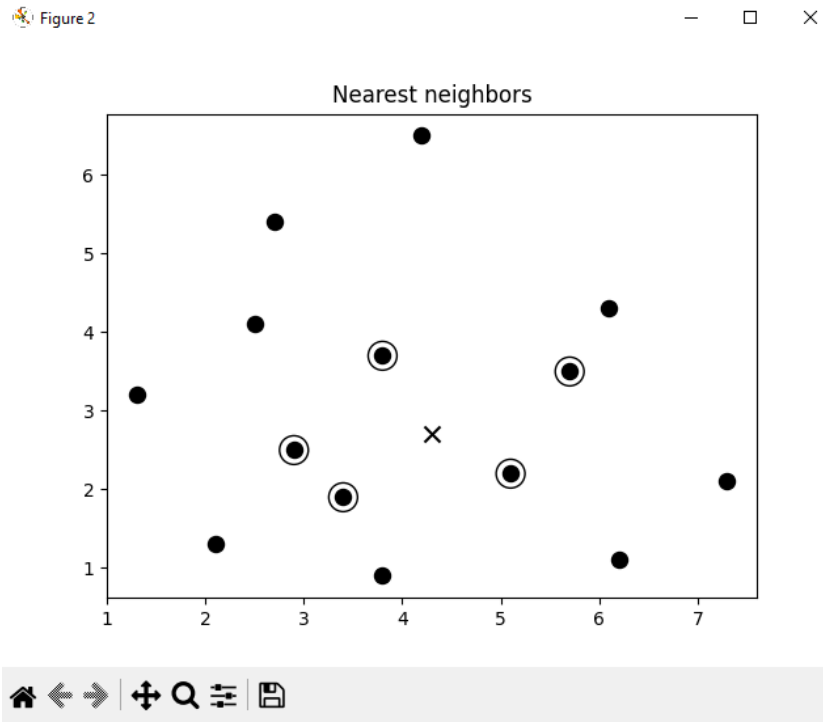
```
from sklearn.datasets import _samples_generator
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.pipeline import Pipeline
from sklearn.ensemble import ExtraTreesClassifier

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

k_best_selector = SelectKBest(f_regression, k=9)

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

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

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

processor_pipeline.fit(X, y)

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

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

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

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

У першому списку представлені результати класифікації для вхідних даних, де кожен елемент є передбаченою категорією для відповідного зразка даних. Загалом є 150 точок даних, кожній з яких призначено клас. Оцінка ефективності класифікатора, що базується на гранично випадковому лісі, визначена значенням Score. Це число відображає точність моделі, де, наприклад, 0.8933 означає, що модель правильно класифікувала близько 89.33% вхідних зразків.

У фінальному рядку наведені індекси ознак, які відіграли ключову роль у класифікації. Ці індекси вказують на конкретні ознаки, які були визнані моделлю як найважливіші для прийняття рішень. У даному випадку було відібрано ознаки з індексами 4, 7, 8, 12, 14, 17 і 22.

Завдання 7:





```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py 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]
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

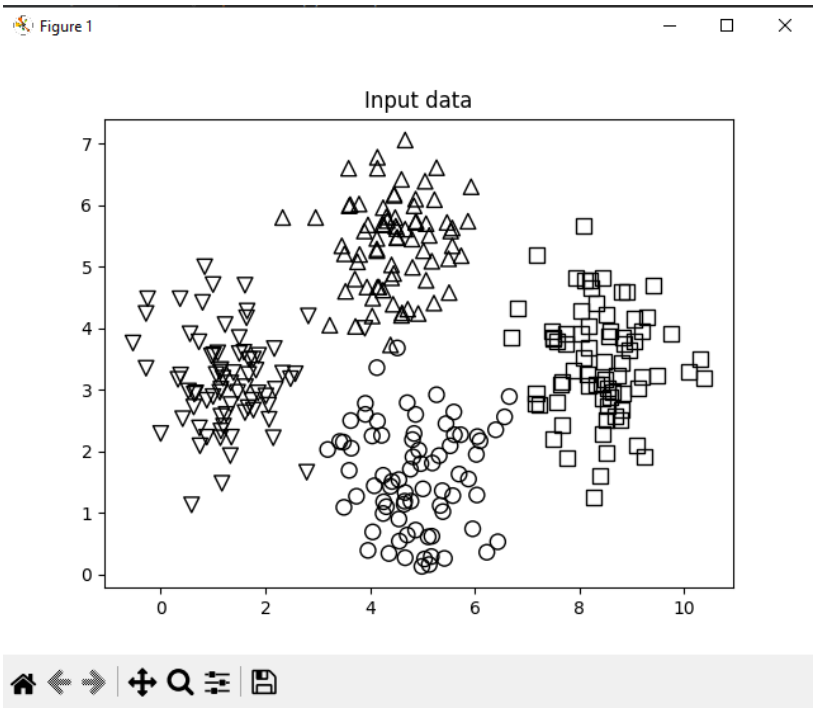
Код програми:

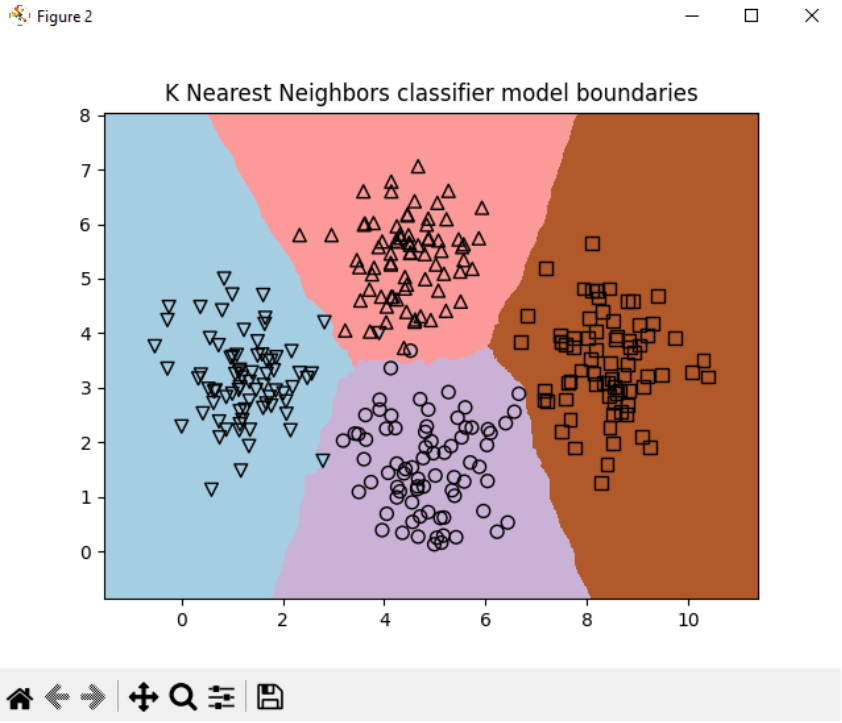
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors

X = np.array([[2.1, 1.3], [1.3, 3.2], [2.9, 2.5], [2.7, 5.4], [3.8, 0.9],
              [7.3, 2.1], [4.2, 6.5], [3.8, 3.7], [2.5, 4.1], [3.4, 1.9],
              [5.7, 3.5], [6.1, 4.3], [5.1, 2.2], [6.2, 1.1]])

k = 5

test_datapoint = [4.3, 2.7]

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

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

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

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

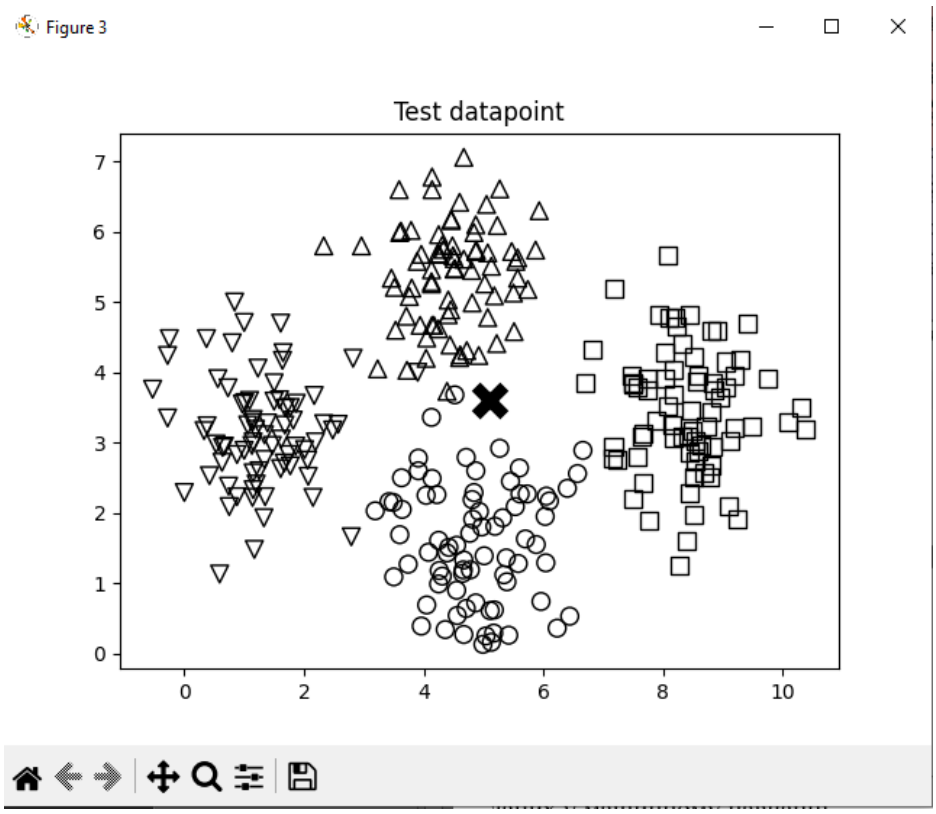
На першому графіку представлені вхідні дані у вигляді двовимірних точок даних. Кожен елемент вибірки позначений чорним кружком, що представляє одну точку даних. Другий графік відображає найближчих сусідів для тестової точки, яку визначено за допомогою методу k найближчих сусідів. Найближчі сусіди відзначені іншими маркерами або кольорами, відмінними від чорних кружків, що відображають вхідні дані.

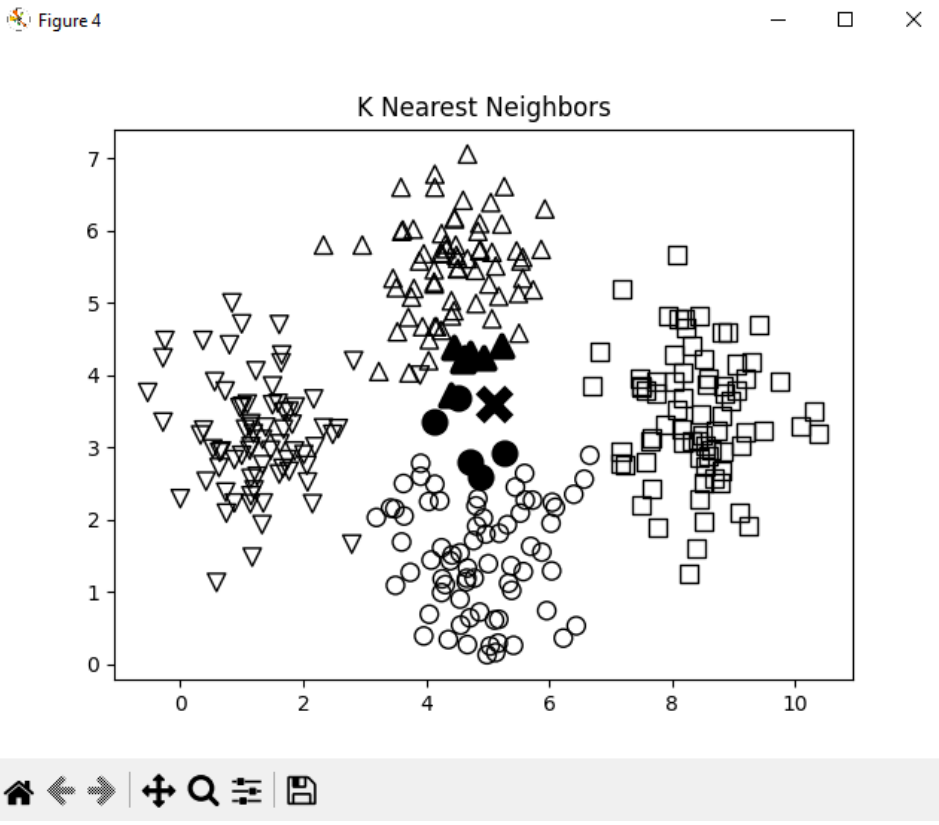
У вікні терміналу надана інформація про найближчих сусідів для тестової точки. Кожен запис містить номер сусіда та його координати, наприклад, "1 ==> [5.1 2.2]", що вказує на те, що перший найближчий сусід має координати (5.1, 2.2). Ця інформація допомагає зрозуміти, які конкретні точки вибірки визнані найближчими до тестової точки за допомогою методу k найближчих сусідів.

Завдання 8:










```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_8.py
Predicted output: 1
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

Код 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].astype(int)

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')

num_neighbors = 12

step_size = 0.01

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

classifier.fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_values, y_values = np.meshgrid(np.arange(x_min, x_max, step_size),
np.arange(y_min, y_max, step_size))

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

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

for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=50, edgecolors='black', facecolors='none')

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

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')

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

```
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()
```

Перший графік представляє вхідні дані, де кожен з чотирьох класів позначений унікальним маркером, вказуючи на клас кожної точки даних. Другий графік відображає розділення між класами, яке було вивчено класифікатором на основі методу k найближчих сусідів. Ця візуалізація показує межі між класами, які модель намагалася вивчити з тренувальних даних.

Третій графік демонструє результати класифікації тестової точки та її k найближчих сусідів. Тестова точка відзначена відмінним маркером чи кольором у порівнянні з вхідними даними, а також позначені k найближчих сусідів, знайдених для цієї точки. У вікні терміналу наведено інформацію про клас, до якого належить тестова точка. Наприклад, у даному випадку тестова точка була класифікована як "1".

Четвертий графік об'єднує всі вхідні дані, карту класифікації, тестову точку та k найближчих сусідів на одному графіку для візуальної оцінки їх взаємного розташування.

Завдання 9:

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Bi
ll Duffy" --score-type Euclidean

Euclidean score:
0.585786437626905
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Bi
ll Duffy" --score-type Pearson

Pearson score:
0.9909924304103233
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Brenda Peterson" --score-type Euclidean

Euclidean score:
0.1424339656566283
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Brenda Peterson" --score-type Pearson

Pearson score:
-0.7236759610155113
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Euclidean

Euclidean score:
0.30383243470068705
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Samuel Miller" --score-type Pearson

Pearson score:
0.7587869106393281
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Euclidean

Euclidean score:
0.2857142857142857
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Julie Hammel" --score-type Pearson

Pearson score:
0
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Euclidean

Euclidean score:
0.28989794855663564
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Clarissa Jackson" --score-type Pearson

Pearson score:
0.6944217062199275
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Adam Cohen" --score-type Euclidean

Euclidean score:
0.38742588672279304
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Adam Cohen" --score-type Pearson

Pearson score:
0.9081082718950217
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Chris Duncan" --score-type Euclidean

Euclidean score:
0.38742588672279304
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_9.py --user1 "David Smith" --user2 "Chris Duncan" --score-type Pearson

Pearson score:
1.0
```

Код програми:

```
import argparse
import json
import numpy as np


def build_arg_parser():
    parser = argparse.ArgumentParser(description='Compute similarity score')
    parser.add_argument('--user1', dest='user1', required=True, help='First
user')
    parser.add_argument('--user2', dest='user2', required=True, help='Second
user')
    parser.add_argument("--score-type", dest="score_type", required=True,
choices=['Euclidean', 'Pearson'],
                        help='Similarity metric to be used')
    return parser


def euclidean_score(dataset,user1,user2):
    if user1 not in dataset:
        raise TypeError('Cannot find ' + user1 + ' in the dataset')

    if user2 not in dataset:
```

```
        raise TypeError('Cannot find ' + user2 + ' in the dataset')

    common_movies = {}

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

    if len(common_movies) == 0:
        return 0

    squared_diff = []

    for item in dataset[user1]:
        if item in dataset[user2]:
            squared_diff.append(np.square(dataset[user1][item] -
dataset[user2][item]))

    return 1 / (1 + np.sqrt(np.sum(squared_diff)))


def pearson_score(dataset, user1, user2):
    if user1 not in dataset:
        raise TypeError('Cannot find ' + user1 + ' in the dataset')

    if user2 not in dataset:
        raise TypeError('Cannot find ' + user2 + ' in the dataset')

    common_movies = {}

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

    num_ratings = len(common_movies)

    if num_ratings == 0:
        return 0

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

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

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

    Sxy = sum_of_products - (user1_sum * user2_sum / num_ratings)
    Sxx = user1_squared_sum - np.square(user1_sum) / num_ratings
    Syy = user2_squared_sum - np.square(user2_sum) / num_ratings

    if Sxx * Syy == 0:
        return 0

    return Sxy / np.sqrt(Sxx * Syy)
```

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

    ratings_file = 'ratings.json'

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

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

Завдання 10:

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py 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 D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

Код 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 in-put 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 = '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', round(float(item[1]), 2))
```

Завдання 11: (легендарні фільми)

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_11.py --user "Chris Duncan"

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

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_11.py --user "Julie Hammel"

Movie recommendations for Julie Hammel:
1. The Apartment
2. Vertigo
3. Raging Bull
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4>
```

```
PS D:\LABS\PRESENT\AI (Python)\Artificial-intelligence\LAB4> py LR_4_task_11.py --user "David Smith"

Movie recommendations for David Smith:
1. Roman Holiday
```

Код програми:

```
import argparse
import json
import numpy as np

from LR_4_task_9 import pearson_score


def build_arg_parser():
    parser = argparse.ArgumentParser(description='Find the movie recommendations
```

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


def get_recommendations(dataset, input_user):
    if input_user not in dataset:
        raise TypeError('Cannot find ' + input_user + ' in the dataset')

    overall_scores = {}
    similarity_scores = {}

    for user in [x for x in dataset if x != input_user]:
        similarity_score = pearson_score(dataset, input_user, user)

        if similarity_score <= 0:
            continue

        filtered_list = [x for x in dataset[user] if x not in \
                         dataset[input_user] or dataset[input_user][x] == 0]

        for item in filtered_list:
            overall_scores.update({item: dataset[user][item] *
similarity_score})
            similarity_scores.update({item: similarity_score})

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

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

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

    movie_recommendations = [movie for _, movie in movie_scores]

    return movie_recommendations


if __name__ == '__main__':
    args = build arg parser().parse args()
    user = args.user

    ratings_file = 'ratings.json'

    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)
```

Висновки: в ході виконання лабораторної роботи було, використовуючи спеціалізовані бібліотеки та мову програмування Python, досліджено методи ансамблів у машинному навчанні та створено рекомендаційні системи.

GitHub: https://github.com/invicibleee/Artificial-intelligence.git