

Кушер Дмитро Євгенович ЗІПЗк-22-1

Лаб 1

Підготовка

Встановив python та всі потрібні бібліотеки

```
Администратор: Командная строка
Microsoft Windows [Version 10.0.22621.2861]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\System32>python -m pip install --upgrade pip

Usage:
  C:\Users\Onibi\AppData\Local\Programs\Python\Python311\python.exe -m pip install [options] <requirement specifier> [package-index-options] ...
  C:\Users\Onibi\AppData\Local\Programs\Python\Python311\python.exe -m pip install [options] -r <requirements file> [package-index-options] ...
  C:\Users\Onibi\AppData\Local\Programs\Python\Python311\python.exe -m pip install [options] [-e] <vcs project url> ...
  C:\Users\Onibi\AppData\Local\Programs\Python\Python311\python.exe -m pip install [options] [-e] <local project path> ...
  C:\Users\Onibi\AppData\Local\Programs\Python\Python311\python.exe -m pip install [options] <archive url/path> ...

no such option: -u

C:\Windows\System32>cd C:\Users\Onibi\AppData\Local\Programs\Python\Python311\Scripts
C:\Users\Onibi\AppData\Local\Programs\Python\Python311\Scripts>pip install numpy scipy matplotlib ipython scikit-learn pandas
Collecting numpy
  Downloading numpy-1.26.2-cp311-cp311-win_amd64.whl (15.8 MB)
----- 15.8/15.8 MB 2.7 MB/s eta 0:00:00
Collecting scipy
  Downloading scipy-1.11.4-cp311-cp311-win_amd64.whl (44.1 MB)
----- 44.1/44.1 MB 4.6 MB/s eta 0:00:00
Collecting matplotlib
  Downloading matplotlib-3.8.2-cp311-cp311-win_amd64.whl (7.6 MB)
----- 7.6/7.6 MB 5.6 MB/s eta 0:00:00
Collecting ipython
  Downloading ipython-8.18.1-py3-none-any.whl (808 kB)
----- 808.2/808.2 kB 5.7 MB/s eta 0:00:00
Collecting scikit-learn
```

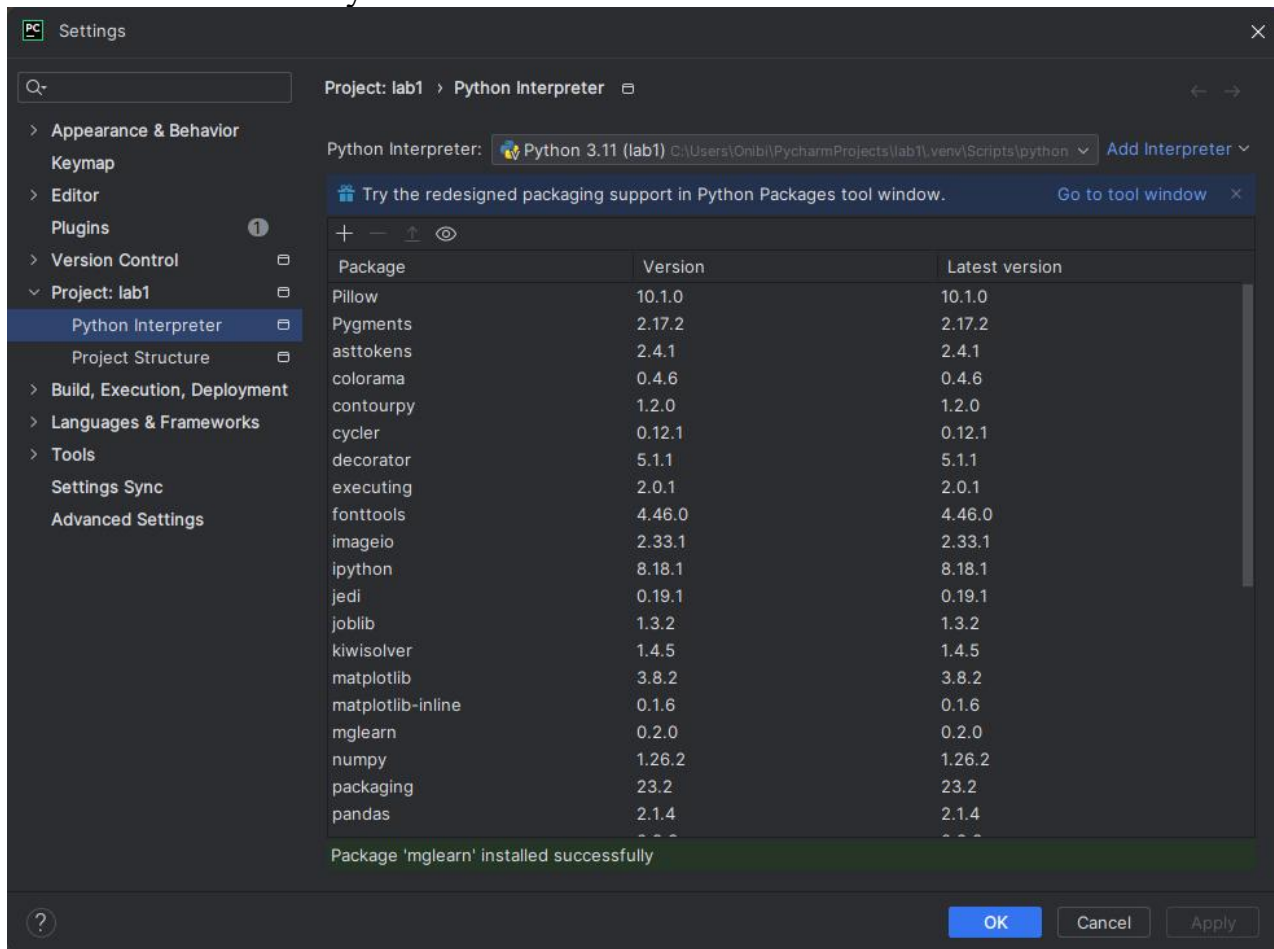
```
Администратор: Командная строка
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Onibi\AppData\Local\Programs\Python\Python311\Scripts>pip install mglearn
Collecting mglearn
  Downloading mglearn-0.2.0-py2.py3-none-any.whl (581 kB)
----- 581.4/581.4 kB 3.3 MB/s eta 0:00:00
Requirement already satisfied: numpy in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (1.26.2)
Requirement already satisfied: matplotlib in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (3.8.2)
Requirement already satisfied: scikit-learn in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (1.3.2)
Requirement already satisfied: pandas in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (2.1.4)
Requirement already satisfied: pillow in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (10.1.0)
Requirement already satisfied: cycler in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (0.12.1)
Collecting imageio
  Downloading imageio-2.33.1-py3-none-any.whl (313 kB)
----- 313.3/313.3 kB 3.3 MB/s eta 0:00:00
Requirement already satisfied: joblib in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from mglearn) (1.3.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib->mglearn) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib->mglearn) (4.46.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib->mglearn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib->mglearn) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib->mglearn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib->mglearn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from pandas->mglearn) (2023.3.post1)
```

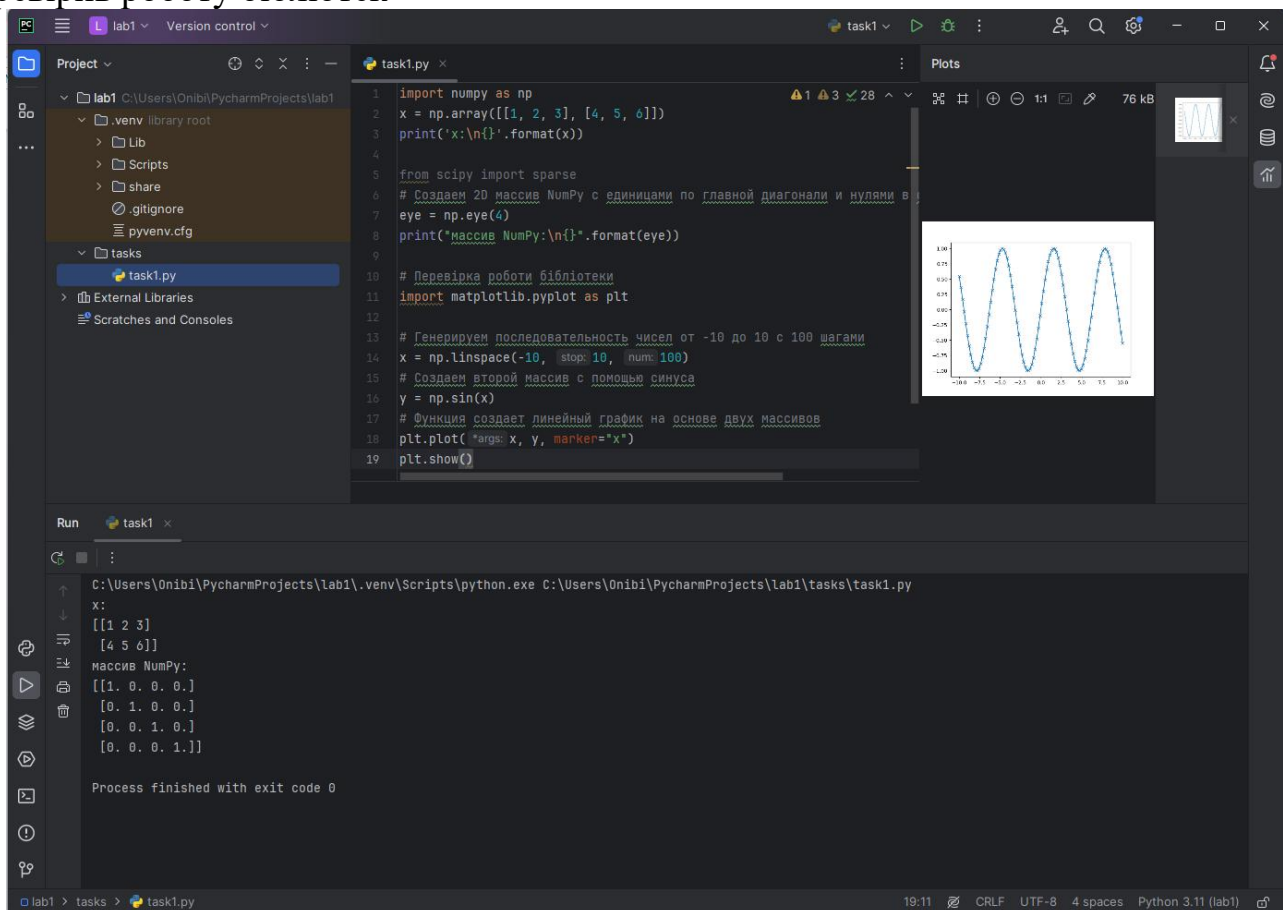
```
Администратор: Командная строка
Successfully installed pip-23.3.1

C:\Users\Onibi\AppData\Local\Programs\Python\Python311\Scripts>pip install seaborn
Collecting seaborn
  Downloading seaborn-0.13.0-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from seaborn) (1.26.2)
Requirement already satisfied: pandas>=1.2 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.3 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from seaborn) (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (4.46.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (10.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.3->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\onibi\appdata\local\programs\python\python311\lib\site-packages (from pandas>=1.2->seaborn) (2023.3)
```

Встановив бібліотеки в Pycharm



Перевірів роботу бібліотек



Завдання 2.1

```
import numpy as np
from sklearn import preprocessing

input_data = np.array([[5.1, -2.9, 3.3],
                        [-1.2, 7.8, -6.1],
                        [3.9, 0.4, 2.1],
                        [7.3, -9.9, -4.5]])

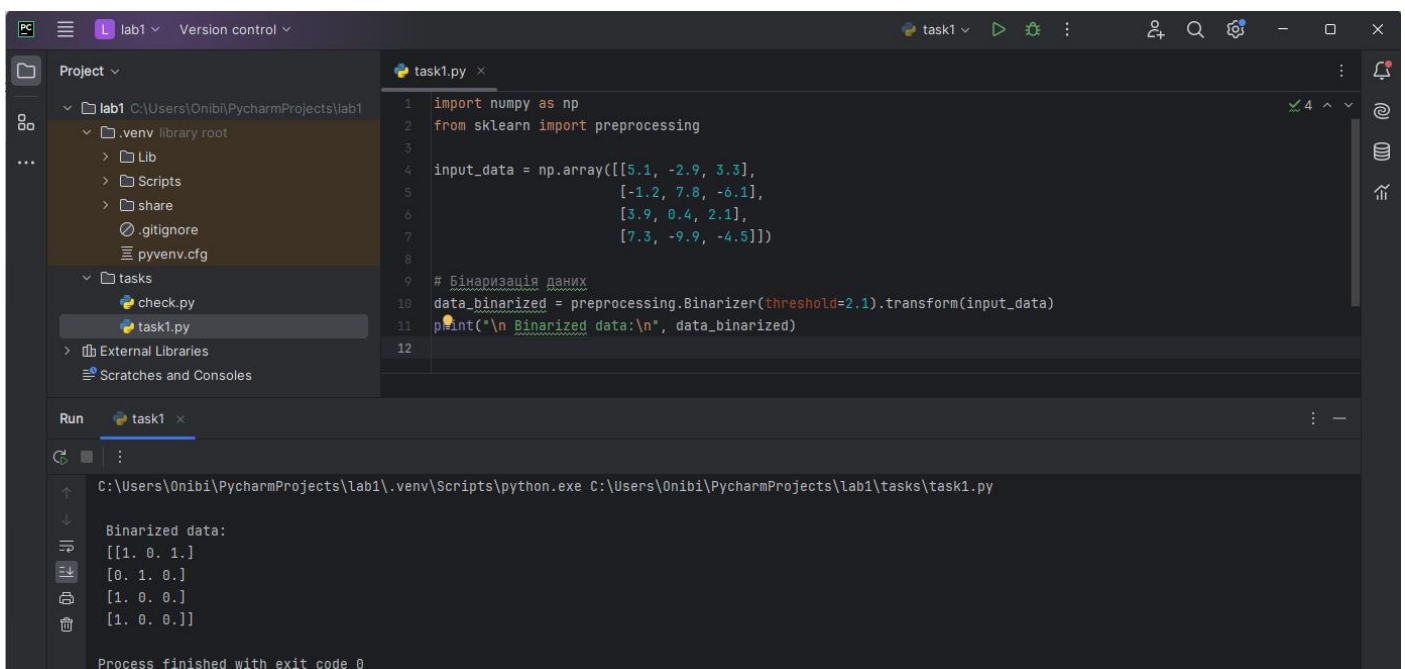
# Бінаризація даних
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\n Binarized data:\n", data_binarized)

# Виведення середнього значення та стандартного відхилення
print("\nBEFORE: ")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

# Исключение среднего
data_scaled = preprocessing.scale(input_data)
print("\nAFTER: ")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))

# Масштабування MinMax
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print("\nMin max scaled data:\n", data_scaled_minmax)

# Нормалізація даних
data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
print("\nl1 normalized data:\n", data_normalized_l1)
print("\nl2 normalized data:\n", data_normalized_l2)
```



The screenshot displays the PyCharm IDE interface. The main editor window shows the Python script from the previous block. The left sidebar shows the project structure with a folder named 'tasks' containing 'task1.py'. The bottom panel shows the 'Run' output, which displays the output of the script's print statements. The output shows the binarized data as a 4x3 array of 0s and 1s.

```
task1.py
1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[5.1, -2.9, 3.3],
5                        [-1.2, 7.8, -6.1],
6                        [3.9, 0.4, 2.1],
7                        [7.3, -9.9, -4.5]])
8
9 # Бінаризація даних
10 data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
11 print("\n Binarized data:\n", data_binarized)
12
```

Run task1

```
C:\Users\0nibi\PycharmProjects\lab1\.venv\Scripts\python.exe C:\Users\0nibi\PycharmProjects\lab1\tasks\task1.py

Binarized data:
[[1. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]

Process finished with exit code 0
```

Виключення середнього, Масштабування та Нормалізація

The screenshot shows the PyCharm IDE with a project named 'lab1'. The file explorer on the left shows the project structure. The main editor displays a Python script 'task1.py' with the following code:

```
1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[5.1, -2.9, 3.3],
5                        [-1.2, 7.8, -6.1],
6                        [3.9, 0.4, 2.1],
7                        [7.3, -9.9, -4.5]])
8
9 # Бінаризація даних
10 data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
11 print("\n Binarized data:\n", data_binarized)
12
13 # Виведення середнього значення та стандартного відхилення
14 print("\nBEFORE: ")
15 print('Mean =', input_data.mean(axis=0))
16 print('Std deviation =', input_data.std(axis=0))
17
18 # Нормалізація даних
19 data_scaled = preprocessing.scale(input_data)
20 print("\nAFTER: ")
21 print('Mean =', data_scaled.mean(axis=0))
22 print('Std deviation =', data_scaled.std(axis=0))
23
24 # Масштабування MinMax
25 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
26 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
27 print("\nMin max scaled data:\n", data_scaled_minmax)
28
29 # Нормалізація даних
30 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
31 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
32 print("\nl1 normalized data:\n", data_normalized_l1)
33 print("\nl2 normalized data:\n", data_normalized_l2)
34
```

The Run console on the right shows the output of the script:

```
C:\Users\Onibi\PycharmProjects\lab1\.venv\Scripts\python.exe C:\Users\Onibi\PycharmProjects\lab1\task1.py
Binarized data:
[[1. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]

BEFORE:
Mean = [ 3.775 -1.15 -1.3 ]
Std deviation = [3.12839661 6.36651396 4.0628192 ]

AFTER:
Mean = [1.11022302e-16 0.00000000e+00 2.77555750e-17]
Std deviation = [1. 1. 1.]

Min max scaled data:
[[0.74117647 0.39548023 1.         ]
 [0.         1.         0.         ]
 [0.6         0.5819209   0.87234043]
 [1.         0.         0.17021277]]

l1 normalized data:
[[0.45132743 -0.25663717 0.2928354 ]
 [-0.0794702  0.51655629 -0.40397351]
 [0.609375    0.0625    0.328125 ]
 [0.33640553 -0.4562212  -0.20737327]]

l2 normalized data:
[[0.75765788 -0.43082507 0.49024922]
 [-0.12030718 0.78199664 -0.61156148]
 [0.87690281 0.08993875 0.47217844]
 [0.55734935 -0.75885734 -0.34357152]]

Process finished with exit code 0
```

Завдання 2.1.5

Кодування міток

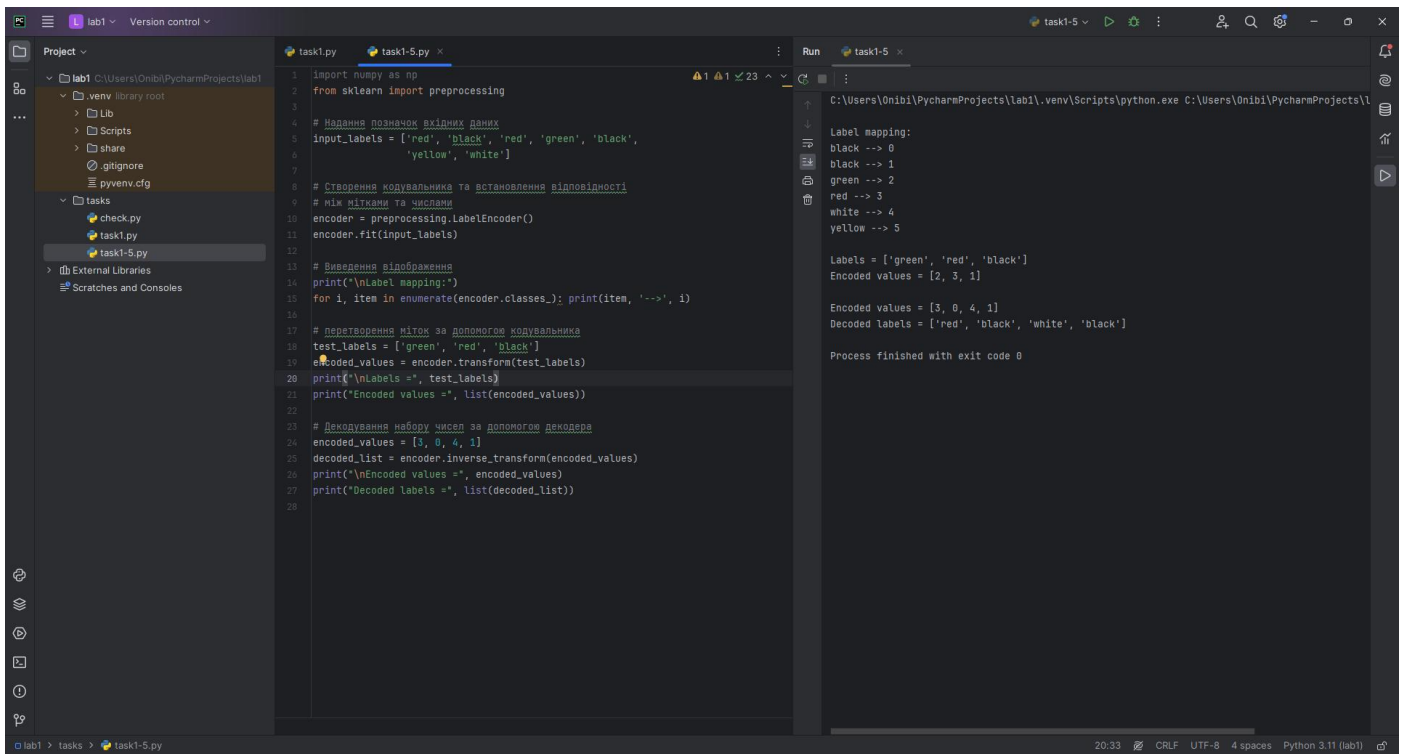
```
import numpy as np
from sklearn import preprocessing
# Надання позначок вхідних даних
input_labels = ['red', 'black', 'red', 'green', 'black',
                'yellow', 'white']

# Створення кодувальника та встановлення відповідності між мітками та числами
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)

# Виведення відображення
print("\nLabel mapping:")
for i, item in enumerate(encoder.classes_): print(item, '-->', i)

# перетворення міток за допомогою кодувальника
test_labels = ['green', 'red', 'black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))

# Декодування набору чисел за допомогою декодера
encoded_values = [3, 0, 4, 1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
print("Decoded labels =", list(decoded_list))
```



Завдання 2.2

Попередня обробка нових даних

```

import numpy as np
from sklearn import preprocessing

input_data = np.array([[ -1.3, 3.9, 4.5],
                        [ -5.3, -4.2, -1.3],
                        [ 5.2, -6.5, -1.1],
                        [ -5.2, 2.6, -2.2]])

# Бінаризація даних
data_binarized = preprocessing.Binarizer(threshold=3).transform(input_data)
print("\n Binarized data:\n", data_binarized)

# Виведення середнього значення та стандартного відхилення
print("\nBEFORE: ")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

# Виключення середнього
data_scaled = preprocessing.scale(input_data)
print("\nAFTER: ")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))

# Масштабування MinMax
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print("\nMin max scaled data:\n", data_scaled_minmax)

# Нормалізація даних
data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')

```

```
print("\nI1 normalized data:\n", data_normalized_l1)
print("\nI2 normalized data:\n", data_normalized_l2)
```

The screenshot shows a PyCharm IDE with a Python script in the main editor and its output in the Run console. The script performs various preprocessing steps on a dataset, including binarization, mean/std deviation calculation, scaling, and normalization.

```
1 import numpy as np
2 from sklearn import preprocessing
3
4 input_data = np.array([[1.3, 3.9, 4.3],
5                        [-5.3, -4.2, -1.3],
6                        [5.2, -0.5, -1.1],
7                        [-5.2, 2.6, -2.2]])
8
9 # Бinarизация данных
10 data_binarized = preprocessing.Binarizer(threshold=5).transform(input_data)
11 print("\n Binarized data:\n", data_binarized)
12
13 # Выведения среднего значения та стандартного відхилення
14 print("\nBEFORE: ")
15 print("Mean =", input_data.mean(axis=0))
16 print("Std deviation =", input_data.std(axis=0))
17
18 # Виключення середнього
19 data_scaled = preprocessing.scale(input_data)
20 print("\nAFTER: ")
21 print("Mean =", data_scaled.mean(axis=0))
22 print("Std deviation =", data_scaled.std(axis=0))
23
24 # Масштабування MinMax
25 data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
26 data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
27 print("\nMin max scaled data:\n", data_scaled_minmax)
28
29 # Нормалізація даних
30 data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
31 data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
32 print("\nI1 normalized data:\n", data_normalized_l1)
33 print("\nI2 normalized data:\n", data_normalized_l2)
34
```

The Run console output shows the results of these operations:

```
Binarized data:
[[0. 1. 1.]
 [0. 0. 0.]
 [1. 0. 0.]
 [0. 0. 0.]]

BEFORE:
Mean = [-1.65 -1.05 -0.025]
Std deviation = [4.27112397 4.40028408 2.64516868]

AFTER:
Mean = [-2.77555756e-17 5.55111512e-17 5.55111512e-17]
Std deviation = [1. 1. 1.]

Min max scaled data:
[[0.38095238 1. 1. ]
 [0. 0.22115385 0.13432836]
 [1. 0. 0.1641791 ]
 [0.08952381 0.875 0. ] ]

I1 normalized data:
[[-0.13402062 0.40206186 0.46391753]
 [-0.49074074 -0.38888889 -0.12037037]
 [ 0.48625 -0.5078125 -0.0859375]
 [-0.52 0.26 -0.22 ]]

I2 normalized data:
[[-0.21328678 0.63986035 0.7383004 ]
 [-0.76965323 -0.60991388 -0.18878287]
 [ 0.61931099 -0.77413873 -0.13100809]
 [-0.83653629 0.41826814 -0.3539192 ]]

Process finished with exit code 0
```

Завдання 2.3

Класифікація логістичною регресією або логістичний класифікатор

```
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
from utilities import visualize_classifier

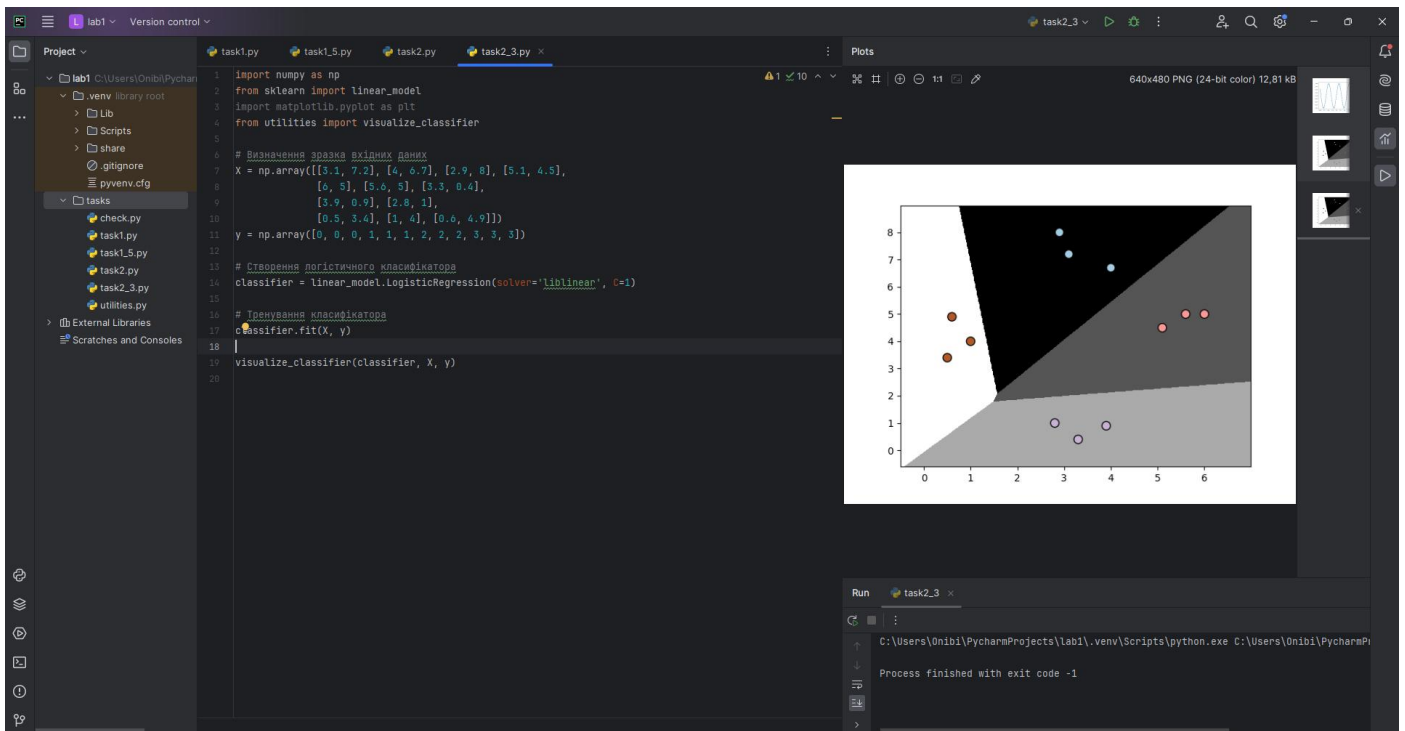
# Визначення зразка вхідних даних
X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5],
              [6, 5], [5.6, 5], [3.3, 0.4],
              [3.9, 0.9], [2.8, 1],
              [0.5, 3.4], [1, 4], [0.6, 4.9]])

y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])

# Створення логістичного класифікатора
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)

# Тренування класифікатора
classifier.fit(X, y)

visualize_classifier(classifier, X, y)
```



Завдання 2.4

Класифікація наївним байєсовським класифікатором

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from utilities import visualize_classifier

# Вхідний файл, який містить дані
input_file = 'data_multivar_nb.txt'

# Завантаження даних із вхідного файлу
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Створення наївного байєсовського класифікатора
classifier = GaussianNB()

# Розбивка даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)

classifier_new = GaussianNB()
classifier_new.fit(X_train, y_train)
y_test_pred = classifier_new.predict(X_test)

# Обчислення якості класифікатора
accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
print("Accuracy of the new classifier =", round(accuracy, 2), "%")

# Візуалізація роботи класифікатора
visualize_classifier(classifier_new, X_test, y_test)

num_folds = 3
accuracy_values = train_test_split.cross_val_score(classifier, X, y, scoring='accuracy',
cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")

```

```

precision_values = train_test_split.cross_val_score(classifier, X, y,
scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = train_test_split.cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = train_test_split.cross_val_score(classifier, X, y, scoring='f1_weighted',
cv=num_folds)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")

```

