

## ЗВІТ

### З лабораторної роботи №2

#### ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ (навчальна дисципліна «СУЧАСНИЙ ШТУЧНИЙ ІНТЕЛЕКТ»)

Студента КН-20-1 навчальної групи

Кірія Даніли Олеговича варіант №6

**Мета:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати.

#### Завдання 1.

Ознаки з набору даних:

Variable Name	Type	Description
age	Integer	Вік
workclass	Categorical	Private, Self-emp-not-inc, Selfemp-inc, Federal-gov, Localgov, State-gov, Without-pay, Never-worked
education	Categorical	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assocacdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
education-num	Integer	Рівень освіти
marital-status	Categorical	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouseabsent, Married-AF-spouse
race	Categorical	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
relationship	Categorical	Wife, Own-child, Husband, Notin-family, Other-relative, Unmarried

occupation	Categorical	Tech-support, Craft-repair, Other-service, Sales, Execmanagerial, Prof-specialty, Handlers-cleaners, Machine-opinspct, Adm-clerical, Farmingfishing, Transport-moving, Privhouse-serv, Protective-serv, Armed-Forces
sex	Binary	Female, Male
capital-gain	Integer	Прибуток
capital-loss	Integer	Витрати
hours-per-week	Integer	Години роботи
native-country	Categorical	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(GuamUSVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad&Tobago, Peru, Hong, Holand-Netherlands
income	Binary	>50K, <=50K

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_score
import warnings

warnings.filterwarnings("ignore")

input_file = 'income_data.txt'

X = []
y = []
count_class1 = 0
count_class2 = 0

```

```

max_datapoints = 25000

with open(input_file, 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue

        data = line[:-1].split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

X = np.array(X)

label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))

classifier.fit(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy',
cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y,
scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, y, scoring='f1_weighted',
cv=num_folds)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")

```

```

input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-
States']

input_data_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = int(input_data[i])
    else:
        input_data_encoded[i] =
int(label_encoder[count].transform([input_data[i]])[0])
        count += 1

input_data_encoded = np.array(input_data_encoded)

predicted_class = classifier.predict([input_data_encoded])
print(label_encoder[-1].inverse_transform(predicted_class)[0])

```

## Код програми

The screenshot shows a code editor with the following Python code for LAB2\_LR\_2\_task\_1.py:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn import preprocessing
4 from sklearn.svm import LinearSVC
5 from sklearn.multiclass import OneVsOneClassifier
6 from sklearn.model_selection import train_test_split, cross_val_score
7 import warnings
8
9 warnings.filterwarnings("ignore")
10
11 input_file = 'income_data.txt'
12
13 X = []
14 y = []
15 count_class1 = 0
16 count_class2 = 0
17 max_datapoints = 25000
18
19 with open(input_file, 'r') as f:
20     for line in f.readlines():
21         if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
22             break
23         if '?' in line:
24             continue
25

```

The Run console at the bottom displays the following output:

```

Run: LAB2_LR_2_task_1
"D:\LABS\PRESENT\AI (Python)\LAB2\venv\Scripts\python.exe" "D:\LABS\PRESENT\AI (Python)\LAB2\LR_2_task_1.py"
Accuracy: 62.64%
Precision: 75.88%
Recall: 62.64%
F1: 56.15%
<=50K
Process finished with exit code 0

```

At the bottom of the console, it also shows: "Packages installed successfully: Installed packages: 'scikit-learn' (5 minutes ago)" and "14/7 CRLF UTF-8 4 spaces Python 3.10 (LAB2)".

## Результат виконання програми

### Завдання 2.

```
62 classifier = SVC(kernel='poly', degree=2, random_state=0)
63
64 classifier.fit(X, y)
```

Run: L\_2\_task\_2 ×

"D:\LABS\PRESENT\AI (Python)\LAB2\venv\Scripts\python.exe" "D:/LABS/PRES  
Accuracy: 77.84%  
Precision: 81.79%  
Recall: 77.84%  
F1: 70.44%  
<=50K  
Process finished with exit code 0

```
62 classifier = OneVsOneClassifier(SVC(kernel='rbf', random_state=0))
63
64 classifier.fit(X, y)
65
for i, item in enumerate(input_... > else
```

Run: L\_2\_task\_2 ×

"D:\LABS\PRESENT\AI (Python)\LAB2\venv\Scripts\python.exe" "D:/LABS/PRESENT/AI (P  
Accuracy: 78.51%  
Precision: 82.93%  
Recall: 78.51%  
F1: 71.65%  
<=50K  
Process finished with exit code 0

```
62 classifier = OneVsOneClassifier(SVC(kernel='sigmoid', random_state=0))
63
64 classifier.fit(X, y)
for i, item in enumerate(input_... > else
```

Run: L\_2\_task\_2 ×

"D:\LABS\PRESENT\AI (Python)\LAB2\venv\Scripts\python.exe" "D:/LABS/PRESENT/AI (Python  
Accuracy: 62.65%  
Precision: 62.63%  
Recall: 62.65%  
F1: 62.64%  
<=50K  
Process finished with exit code 0

### Завдання 3.

```
import numpy as np
from sklearn.datasets import load_iris

iris_dataset = load_iris()

print("Ключі iris_dataset: \n{}\n".format(iris_dataset.keys()))
print(iris_dataset['DESCR'][:193] + "\n...")
print("Назви відповідей: {}".format(iris_dataset['target_names']))
print("Назва ознак: \n{}".format(iris_dataset['feature_names']))
print("Тип масиву data: {}".format(type(iris_dataset['data'])))
print("Форма масиву data: {}".format(iris_dataset['data'].shape))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}\n".format(iris_dataset['target']))

print(format(iris_dataset['feature_names']))
for i in range(5):
    print('{}'.format(i+1) + ") " + '{}'.format(iris_dataset['data'][i]))
```

## Код програми

[illegible]

## Результат виконання програми

```

from pandas import read_csv
from pandas.plotting import scatter_matrix
import matplotlib
import matplotlib.pyplot as pyplot
from sklearn.model_selection import train_test_split

matplotlib.use('TkAgg')

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)

print(dataset.shape)
print(dataset.head(20))
print(dataset.describe())
print(dataset.groupby('class').size())

dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False,
sharey=False)
pyplot.show()
dataset.hist()
pyplot.show()

array = dataset.values
X = array[:, 0:4]
y = array[:, 4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)

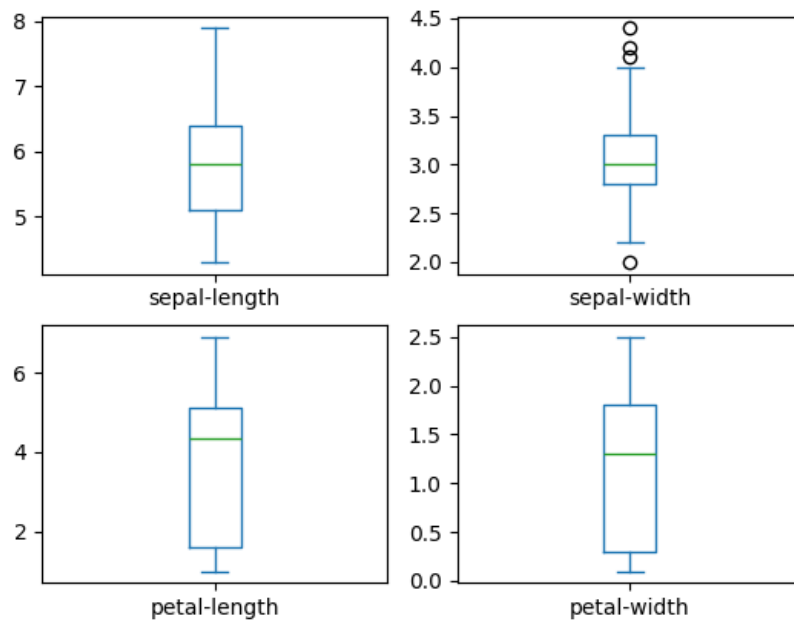
scatter_matrix(dataset)
pyplot.show()

```

Код програми (1 – 3)

Figure 1

— □ ×

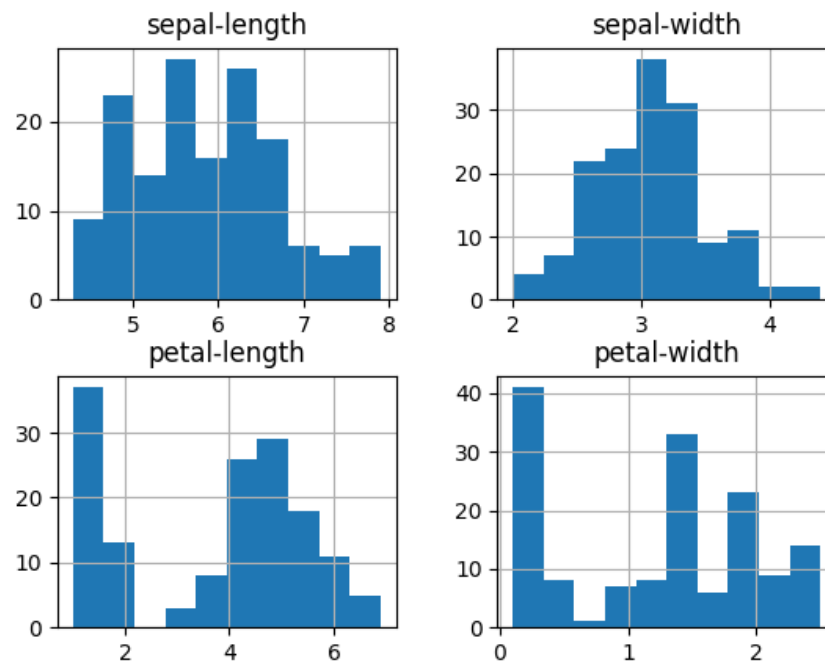


Home Left Right Pan Zoom Fit Save

x= y=5.74

Figure 1

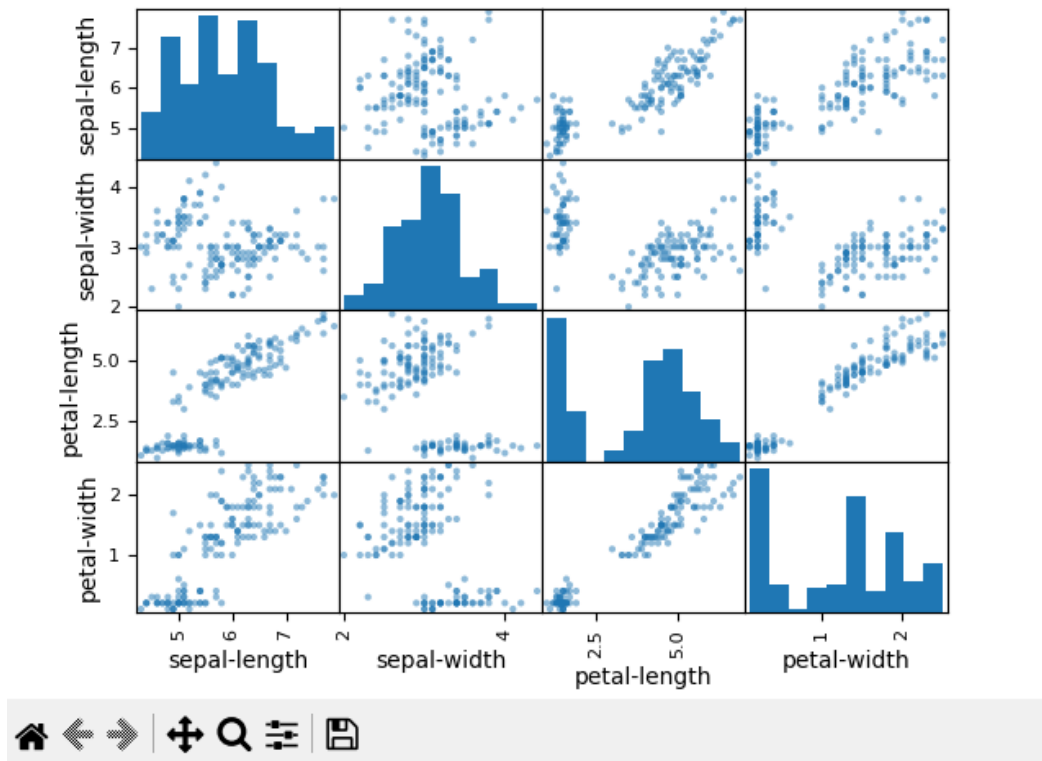
— □ ×



Home Left Right Pan Zoom Fit Save



Figure 1



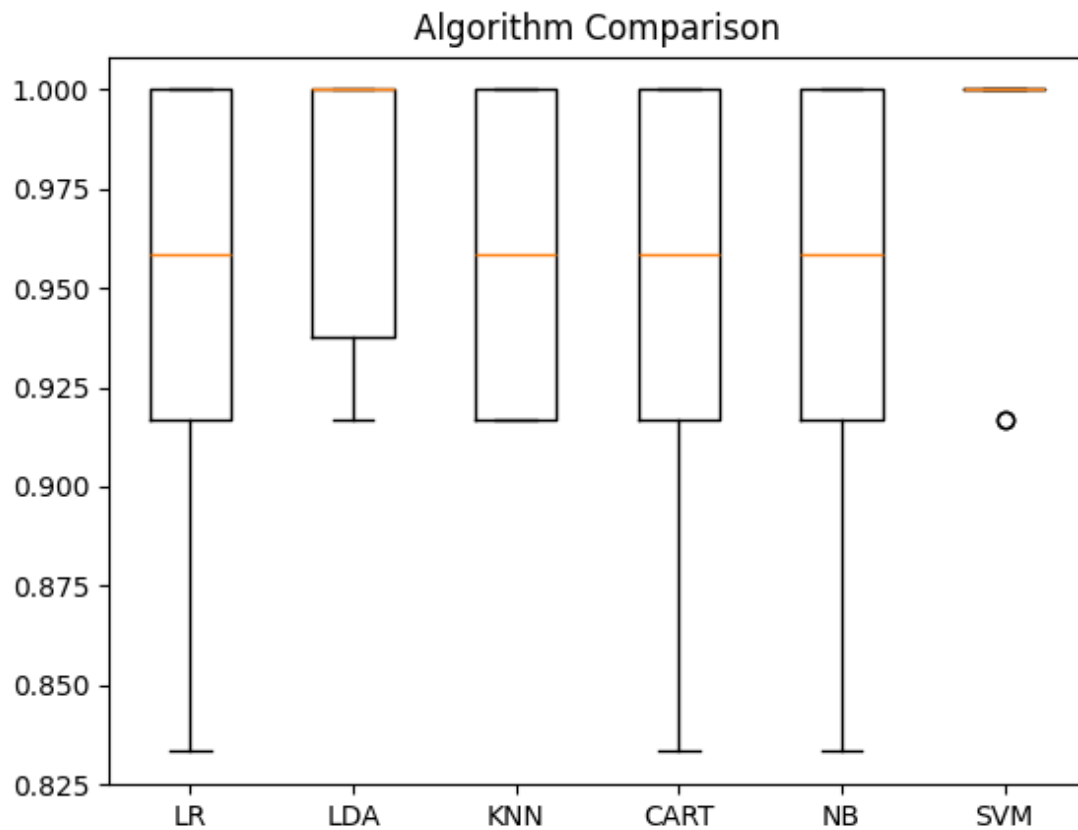
```
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold,
    scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

Код програми (4)

Figure 1



Run: L\_3\_task\_3\_2 x

```

count      150.000000    150.000000    150.000000    150.000000
mean        5.843333      3.054000      3.758667      1.198667
std         0.828066      0.433594      1.764420      0.763161
min         4.300000      2.000000      1.000000      0.100000
25%         5.100000      2.800000      1.600000      0.300000
50%         5.800000      3.000000      4.350000      1.300000
75%         6.400000      3.300000      5.100000      1.800000
max         7.900000      4.400000      6.900000      2.500000

class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)

```

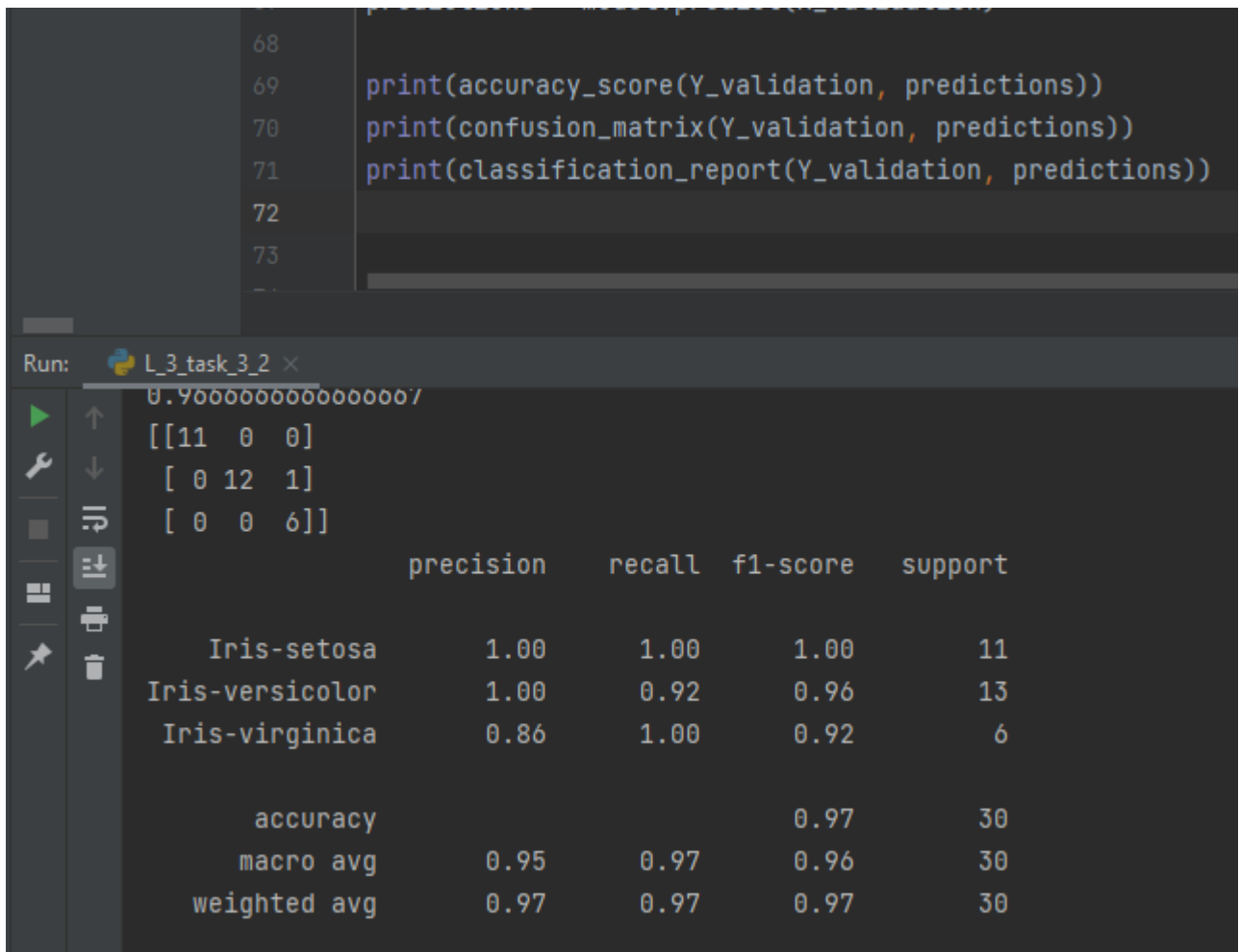
Run | TODO | Problems | Terminal | Python Packages | Python Console

Серед методів класифікації, розглянутих у нашому дослідженні, метод опорних векторів (SVM) виявився найбільш оптимальним. Цей метод продемонстрував найвищу середню точність та мінімальне стандартне відхилення в порівнянні з іншими методами. Крім того, слід відзначити, що метод опорних векторів відомий своєю здатністю ефективно працювати з різними типами даних і успішно справлятися з складними завданнями класифікації.

```
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

Код програми (6-7)



```
68
69     print(accuracy_score(Y_validation, predictions))
70     print(confusion_matrix(Y_validation, predictions))
71     print(classification_report(Y_validation, predictions))
72
73
```

Run: L\_3\_task\_3\_2 ×

0.9000000000000001

```
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.92	0.96	13
Iris-virginica	0.86	1.00	0.92	6
accuracy			0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

```
import numpy as np
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма X_new:", X_new.shape)
```

```
prediction = model.predict(X_new)
print("Прогноз: {}".format(prediction))
predicted_class = prediction[0]
print("Мітка: {}".format(predicted_class))
```

#### Код програми (8)

```
Форма X_new: (1, 4)
Прогноз: ['Iris-setosa']
Мітка: Iris-setosa

Process finished with exit code 0
```

На підставі аналізу точності, матриці помилок та звіту про класифікацію можна зробити припущення, що досягнута висока ефективність класифікації на наборі даних Iris. За результатами аналізу можна стверджувати, що квітка, представлена восьмим прикладом, відноситься до виду Iris-setosa.

#### Завдання 4.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

input_file = 'income_data.txt'
dataset = pd.read_csv(input_file, sep=',', header=None, names=[
    'Age', 'Workclass', 'fnlwgt', 'Education', 'Education_Num',
    'Marital_Status',
    'Occupation', 'Relationship', 'Race', 'Sex', 'Capital_Gain', 'Capital_Loss',
    'Hours_Per_Week', 'Native_Country', 'Income'
])

dataset_encoded = pd.get_dummies(dataset, columns=[
    'Workclass', 'Education', 'Marital_Status',
    'Occupation', 'Relationship', 'Race', 'Sex', 'Native_Country'
])

X = dataset_encoded.drop('Income', axis=1)
```

```

y = dataset_encoded['Income']

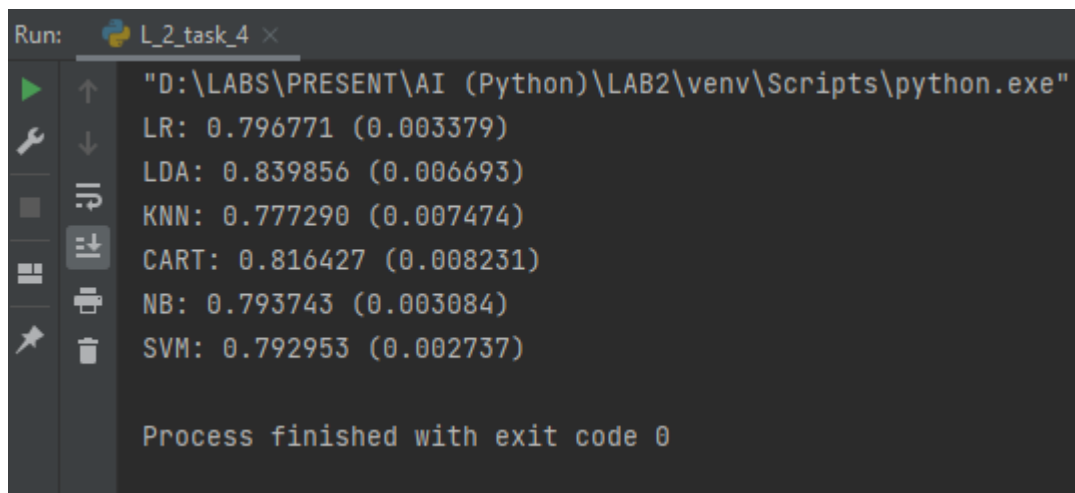
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='scale'))))

results = []
names = []
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

```

### Код програми



```

Run: L_2_task_4 ×
"D:\LABS\PRESENT\AI (Python)\LAB2\venv\Scripts\python.exe"
LR: 0.796771 (0.003379)
LDA: 0.839856 (0.006693)
KNN: 0.777290 (0.007474)
CART: 0.816427 (0.008231)
NB: 0.793743 (0.003084)
SVM: 0.792953 (0.002737)

Process finished with exit code 0

```

За отриманими результатами можна визначити, що в даному контексті найкращою опцією є метод аналізу лінійних дискримінантів (LDA). Ця модель ефективно враховує взаємозв'язок між ознаками та цільовою змінною. Крім того, стандартне відхилення для LDA є низьким, що свідчить про стабільність цієї моделі, і, важливо відзначити, що її розрахунок не потребує значних обчислювальних ресурсів.

### Завдання 5.

```

import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import RidgeClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from io import BytesIO
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

matplotlib.use('TkAgg')

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(X_train, y_train)

ypred = clf.predict(X_test)

print('Accuracy:', np.round(metrics.accuracy_score(y_test, ypred), 4))
print('Precision:', np.round(metrics.precision_score(y_test, ypred,
average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(y_test, ypred,
average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(y_test, ypred, average='weighted'),
4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(y_test, ypred),
4))
print('Matthews Corrocoef:', np.round(metrics.matthews_corrcoef(y_test, ypred),
4))
print('\t\tClassification Report:\n', metrics.classification_report(ypred,
y_test))

mat = confusion_matrix(y_test, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.savefig("Confusion.jpg")

f = BytesIO()
plt.savefig(f, format="svg")

plt.savefig("Confusion.svg")

```

Код программы

Run: L\_2\_task\_5

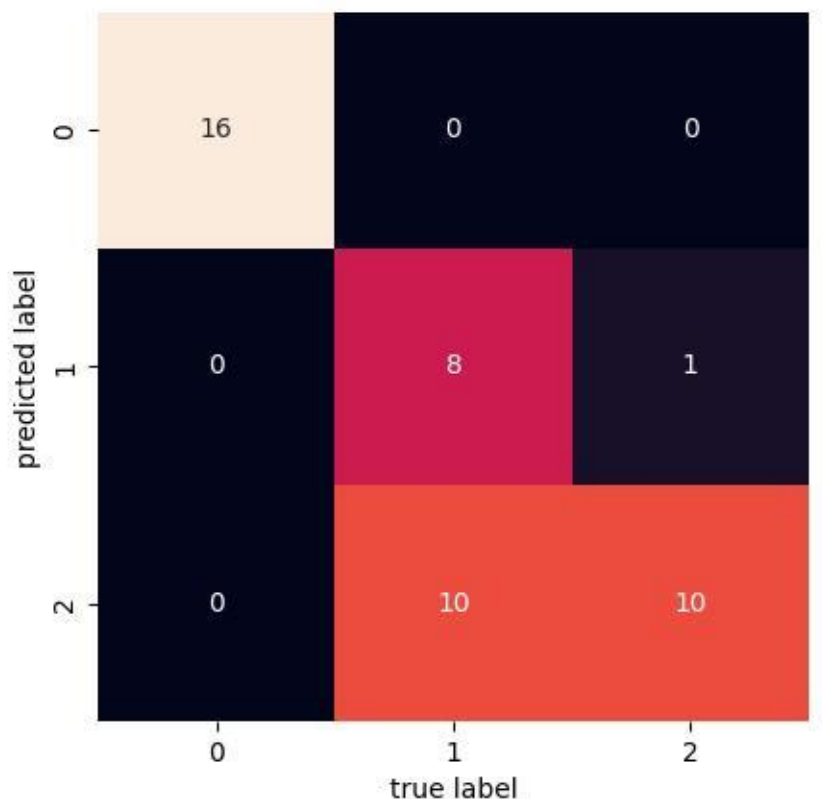
```
"D:\LABS\PRESENT\AI (Python)\LAB2\venv\Scripts\python.exe" "D:/LABS
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831

Classification Report:
              precision    recall  f1-score   support

     0              1.00      1.00      1.00        16
     1              0.44      0.89      0.59         9
     2              0.91      0.50      0.65        20

 accuracy              0.76              45
 macro avg              0.78      0.80      0.75        45
weighted avg              0.85      0.76      0.76        45

Process finished with exit code 0
```



Метод Ridge використовується для класифікації даних і має параметри, які впливають на його роботу. Параметр `"tol=1e-2"` визначає точність обчислення і використовується у випадках, коли оптимізаційний алгоритм повинен завершити свою роботу. Параметр `"solver='sag'"` вказує на метод оптимізації для класифікатора Ridge, де `"sag"` означає Stochastic Average Gradient Descent.

Щодо показників якості:

- Accuracy визначає відсоток правильних передбачень у порівнянні з загальною кількістю прикладів, і в цьому випадку вона становить 75%.
- Precision вимірює точність передбачень позитивних класів і складає 83%.
- Recall вимірює здатність моделі виявляти всі позитивні приклади, і його значення також становить 75%.
- F1 Score представляє гармонічне середнє між точністю і повнотою, і в даному випадку він становить 75%.

Матриця заплутаності `"Confusion.jpg"` вказує на кількість правильних і неправильних класифікацій для кожного класу. На головній діагоналі знаходиться кількість правильних класифікацій для кожного класу, і поза



головною діагоналлю показана кількість неправильних класифікацій для кожного класу.

Коефіцієнт Коена Каппа (Cohen Kappa Score) вимірює ступінь узгодженості між реальними та передбаченими мітками, враховуючи можливість випадкового вибору. Значення Каппа у цьому випадку становить 0.6431, що вказує на помірний рівень узгодженості між фактичними та передбаченими класами.

Коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient) подібний до Коефіцієнта Коена Каппа, але більше враховує дисбаланс класів у вибірці. Зазвичай використовується для бінарної класифікації, і в даному випадку його значення становить 0.6831, що свідчить про добрий рівень узгодженості між фактичними та передбаченими класами.

**Висновки:** в ході виконання лабораторної роботи було, використовуючи спеціалізовані бібліотеки та мову програмування Python, досліджено різні методи класифікації даних та отримано практичні навички в їх порівнянні.

GitHub: <https://github.com/invincibleee/Artificial-intelligence.git>