

ЛАБОРАТОРНА РОБОТА № 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

Хід роботи:

Завдання 2.1. Класифікація за допомогою машин опорних векторів (SVM)

Числові ознаки: age, flnwgwt, education-num, capital-gain, capital-loss, hours-per-week;
категоріальні: workclass, education, marital-status, occupation, relationship, race, sex, native-country.

Лістинг програми:

```
4 import numpy as np
3 import matplotlib.pyplot as plt      ■ "plt" is not accessed
2 from sklearn import preprocessing, metrics  ■ "metrics" is not accessed
1 from sklearn.svm import LinearSVC
5 from sklearn.multiclass import OneVsOneClassifier
1 from sklearn.model_selection import cross_val_score, train_test_split

2
3
4 input_file = 'income_data.txt'
5 X = []
6 y = []
7 count_class1 = 0
8 count_class2 = 0
9 max_datapoints = 30000
10 with open(input_file, 'r') as f:
11     for line in f.readlines():
12         if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
13             break
14         if '?' in line:
15             data = line[:-1].split(',')
16             if data[-1] == '<=50K' and count_class1 < max_datapoints:
17                 X.append(data)
18                 count_class1 += 1
19
20             if data[-1] == '>50K' and count_class2 < max_datapoints:
21                 X.append(data)
22                 count_class2 += 1
23
24
25 X = np.array(X)
26 label_encoder = []
27 X_encoded = np.empty(X.shape)
28 for i, item in enumerate(X[0]):
29     if item.isdigit():
30         X_encoded[:, i] = X[:, i]
31     else:
32         label_encoder.append(preprocessing.LabelEncoder())
33         X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
34
35 X = X_encoded[:, :-1].astype(int)
36 y = X_encoded[:, -1].astype(int)
37 classifier = OneVsOneClassifier(LinearSVC(random_state=0))
38 classifier.fit(X, y)
39 X_train, X_test, y_train, y_test = train_test_split(
40     X, y, test_size=0.2, random_state=5)
```

					«Житомирська політехніка».23.123.03.000 – Лр2			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Бабіч Д. В.			Звіт з лабораторної роботи	Лім.	Арк.	Аркушіє
Перевір.		Пулеко І. В.					1	7
Керівник						ФІКТ Гр. КІм-22-1		
Н. контр.								
Зав. каф.								

```

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

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:
        tmp = []
        tmp.append(input_data[i])
        input_data_encoded[i] = int(label_encoder[count].transform(tmp))
        count += 1
        tmp = []

input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
predicted_class = classifier.predict(input_data_encoded)
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)) + "%")
print('Predicted class is: ',
      label_encoder[-1].inverse_transform(predicted_class)[0])

```

		Бабич Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пудеко І. В.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат роботи програми:

```
AI/lab2 on ʘ main [?]  
→ python3 -W ignore LR_2_task1.py  
Accuracy: 64.55%  
Precision: 86.34%  
Recall: 64.55%  
F1: 59.63%  
Predicted class is: <=50K
```

Тестова точка належить до класу '<=50K'.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Результати роботи класифікаторів:

```
AI/lab2 on ʘ main [?]  
→ python3 -W ignore LR_2_task2_1.py  
=====Poly kernel=====
```

Accuracy: 86.41%
Precision: 85.43%
Recall: 86.41%
F1: 80.52%
Predicted class is: <=50K

```
AI/lab2 on ʘ main [?]  
→ python3 -W ignore LR_2_task2_2.py  
=====Gaussian kernel=====
```

Accuracy: 86.58%
Precision: 88.39%
Recall: 86.58%
F1: 80.77%
Predicted class is: <=50K

```
AI/lab2 on ʘ main [?]  
→ python3 -W ignore LR_2_task2_3.py  
=====Sigmoid kernel=====
```

Accuracy: 86.33%
Precision: 83.58%
Recall: 86.33%
F1: 80.19%
Predicted class is: <=50K

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Код програми:

```
27 url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
26 names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
25 dataset = read_csv(url, names=names)
24
23 array = dataset.values
22
21 X = array[:, 0:4]
20 y = array[:, 4]
19 X_train, X_validation, Y_train, Y_validation = train_test_split(
18     X, y, test_size=0.20, random_state=1)
17 models = []
16 models.append(('LR', LogisticRegression(
15     solver='liblinear', multi_class='ovr'))))
14 models.append(('LDA', LinearDiscriminantAnalysis()))
13 models.append(('KNN', KNeighborsClassifier()))
12 models.append(('CART', DecisionTreeClassifier()))
11 models.append(('NB', GaussianNB()))
10 models.append(('SVM', SVC(gamma='auto'))))
9 results = []
8 names = []
7
6 for name, model in models:
5     kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
4     cv_results = cross_val_score(
3         model, X_train, Y_train, cv=kfold, scoring='accuracy')
2     results.append(cv_results)
1     names.append(name)
08 print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
1 # pyplot.boxplot(results, labels=names)
2 # pyplot.title('Algorithm Comparison')
3 # pyplot.show()
4
5 model = SVC(gamma='auto')
6 model.fit(X_train, Y_train)
7 predictions = model.predict(X_validation)
8 # print(accuracy_score(Y_validation, predictions))
9 # print(confusion_matrix(Y_validation, predictions))
10 # print(classification_report(Y_validation, predictions))
11
12
13 X_new = np.array([[5, 2.9]]) # Argument of type "list[list[int | float]]" cannot be assigned to para
14 knn = KNeighborsClassifier(n_neighbors=5)
15 x_train, x_test, y_train, y_test = train_test_split(iris_dataset.data[:, 2:4], # Cannot access membe
16     iris_dataset['target'], # Argument of type "Lite
17     random_state=0)
18 knn_model = knn.fit(x_train, y_train)
19 prediction = knn.predict(X_new)
20 predictions = knn.predict(x_test)
21 accuracy = accuracy_score(y_test, predictions)
22 print(f'Accuracy: {accuracy}')
23 print("forecast: {}".format(iris_dataset['target_names'][prediction[0]])) # Argument of type "Litera
```

Результат виконання програми:

		Бабіч Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пулеко І. В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```
→ python3 -W ignore LR_2_task3.py
iris_dataset's keys: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:
```

Iris plants dataset

****Data Set Characteristics:****

```
:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm
  - class:
    - Iris-Setosa
    - Iris-Versicolour
    - Iris-Virginica
```

:Summary Statistics:

	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

```
:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988
```

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

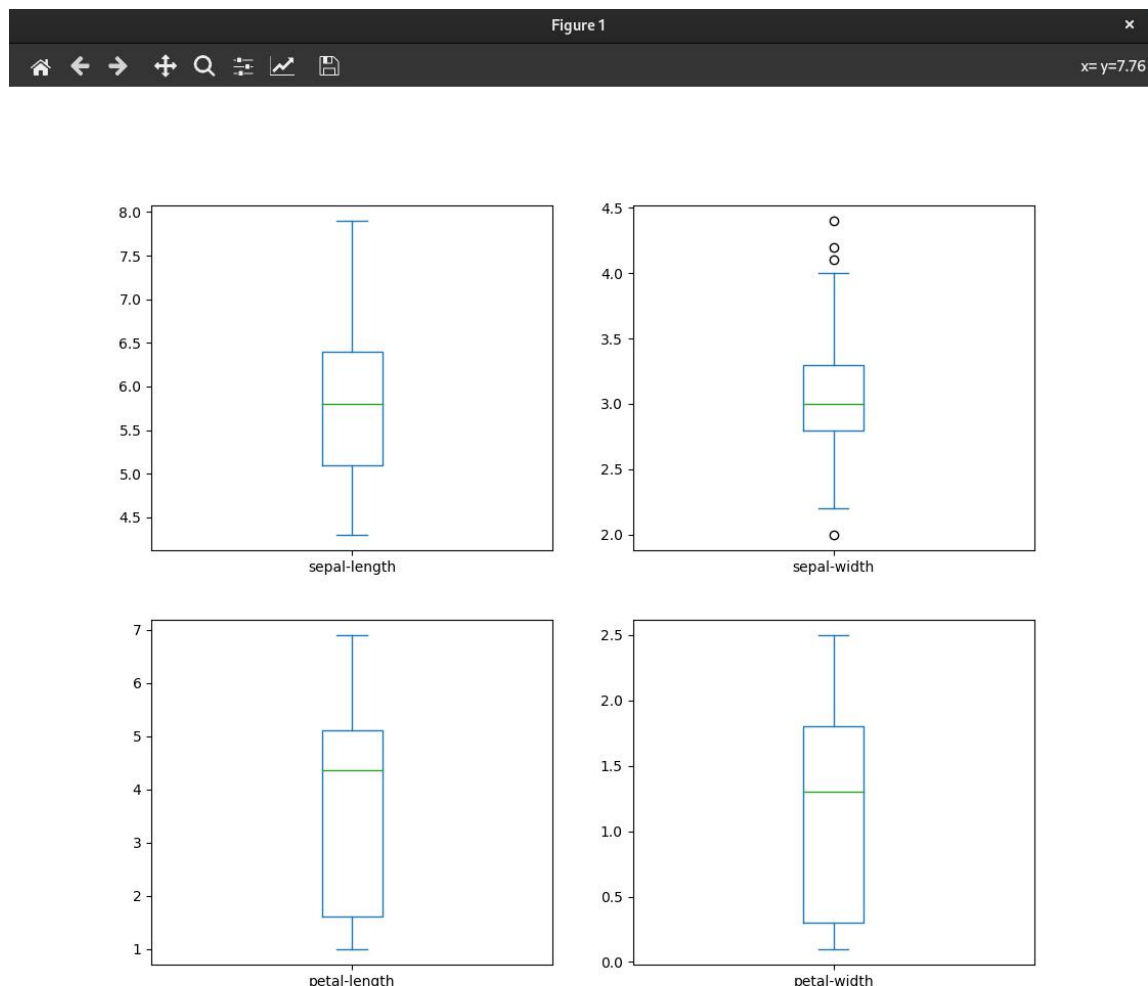
.. topic:: References

```
- Fisher, R.A. "The use of multiple measurements in taxonomic problems"
  Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
```

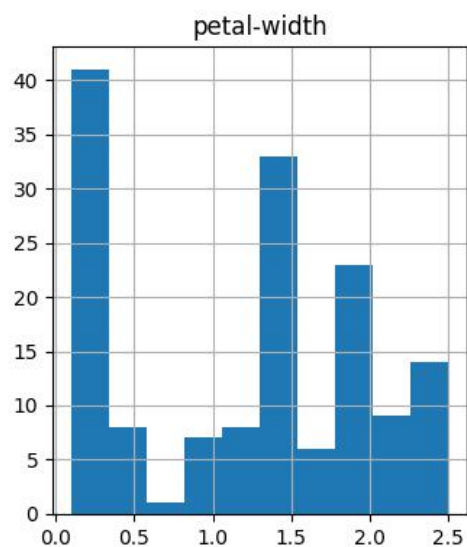
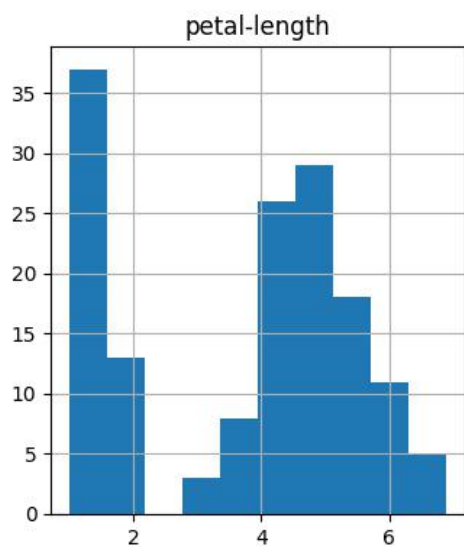
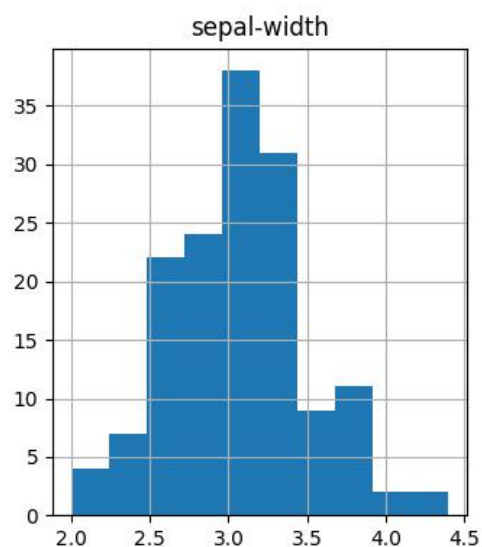
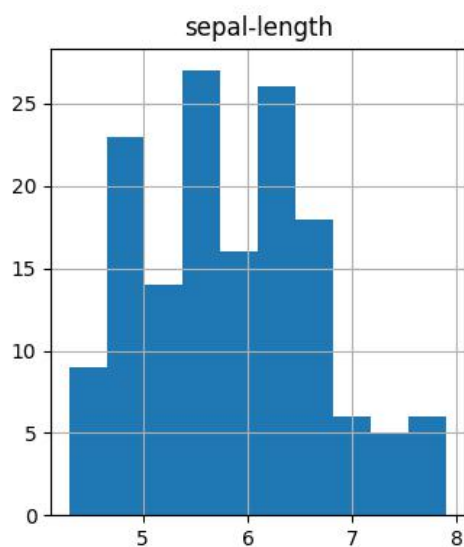
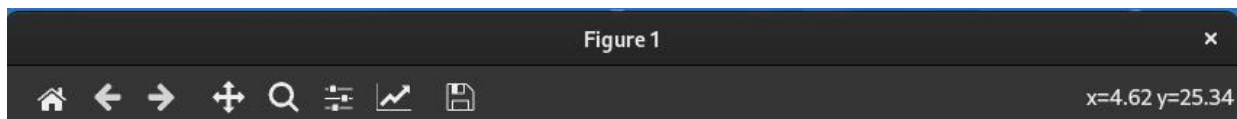
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's AUTOCLASS II conceptual clustering system finds 3 classes in the data.
- Many, many more ...

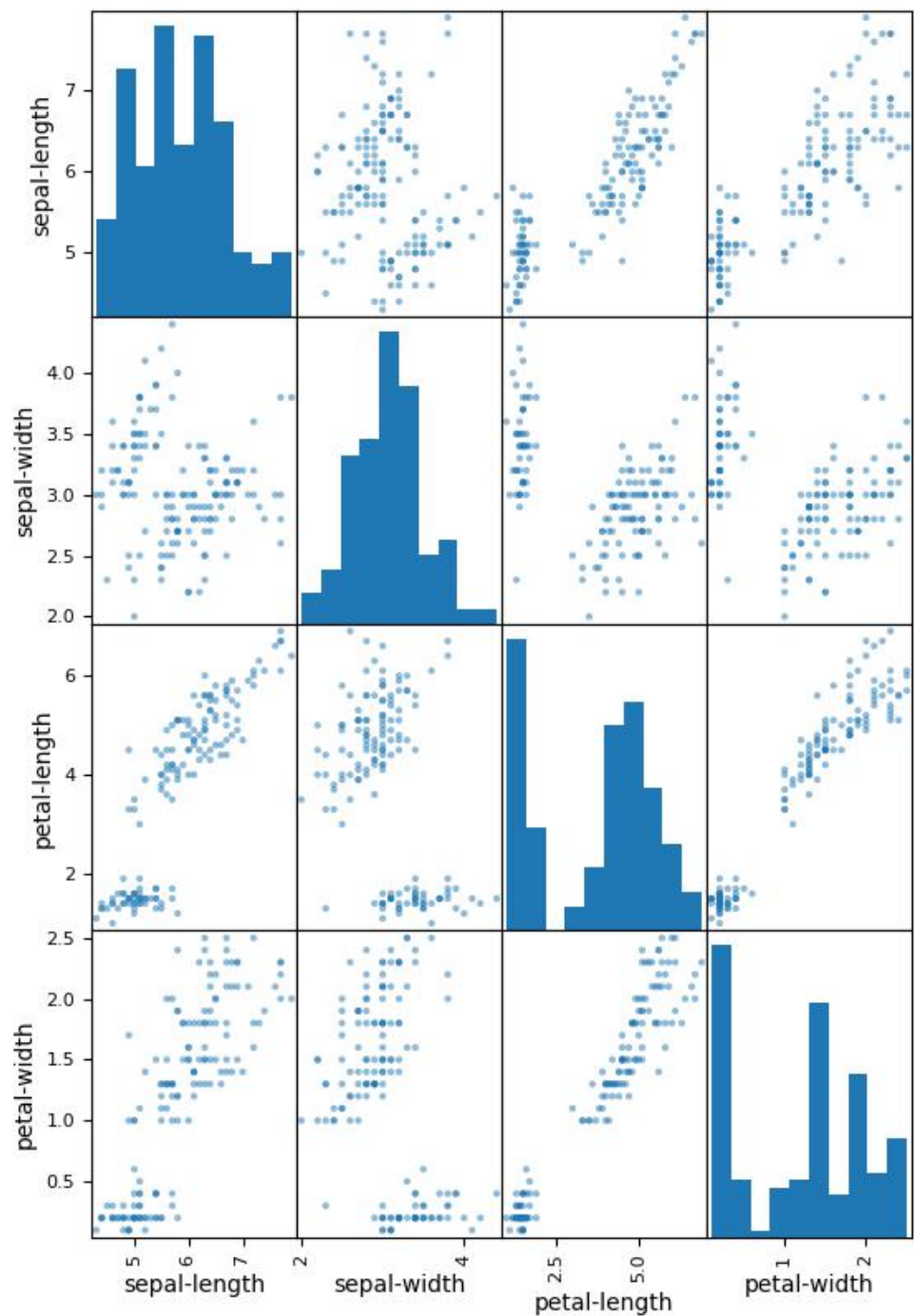
```
...
target names: ['setosa' 'versicolor' 'virginica']
feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
type of data array: <class 'numpy.ndarray'>
shape of data array: (5, 4)
type of target array: <class 'numpy.ndarray'>
targets: [0 0 0 0 0]
Accuracy: 0.9736842105263158
forecast: virginica
```

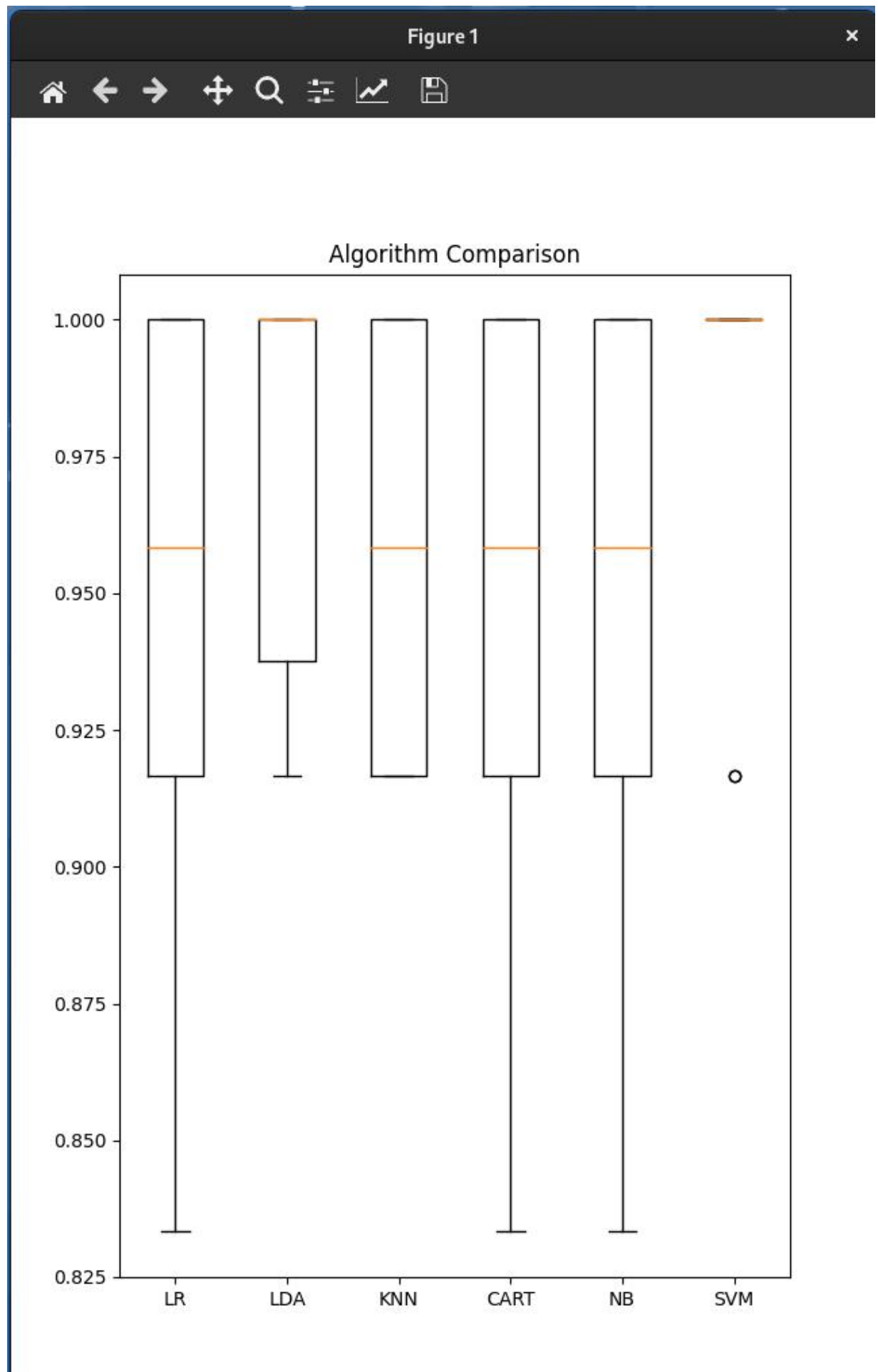
Побудовані графіки:



		Бабіч Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пулеко І. В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		







Результат роботи програми:

```

targets: [0 0 0 0 0]
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.055277)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
0.9666666666666667
[[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

Accuracy: 0.9736842105263158
forecast: virginica

```

За результатами тренування вдалося досягти точності 0.97 та встановити, що квітка належить до класу virginica.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

		Бабіч Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пудеко І. В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Код програми:

```

23 from matplotlib import pyplot
24 from sklearn import preprocessing
25 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
26 from sklearn.linear_model import LogisticRegression
27 from sklearn.model_selection import StratifiedKFold, cross_val_score, train_test_split
28 from sklearn.naive_bayes import GaussianNB
29 from sklearn.neighbors import KNeighborsClassifier
30 from sklearn.svm import SVC
31 from sklearn.tree import DecisionTreeClassifier
32 import numpy as np
33
34 input_file = 'income_data.txt'
35 X = []
36 y = []
37 count_class1 = 0
38 count_class2 = 0
39 max_datapoints = 30000
40 with open(input_file, 'r') as f:
41     for line in f.readlines():
42         if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
43             break
44         if '?' in line:
45             data = line[:-1].split(',')
46             if data[-1] == '<=50K' and count_class1 < max_datapoints:
47                 X.append(data)
48                 count_class1 += 1
49             if data[-1] == '>50K' and count_class2 < max_datapoints:
50                 X.append(data)
51                 count_class2 += 1
52
53 X = np.array(X)
54 label_encoder = []
55 X_encoded = np.empty(X.shape)
56 for i, item in enumerate(X[0]):
57     if item.isdigit():
58         X_encoded[:, i] = X[:, i]
59     else:
60         label_encoder.append(preprocessing.LabelEncoder())
61         X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
62
63 X = X_encoded[:, :-1].astype(int)
64 y = X_encoded[:, -1].astype(int)
65
66 X_train, X_validation, Y_train, Y_validation = train_test_split(
67     X, y, test_size=0.20, random_state=1)
68 models = []
69 models.append(('LR', LogisticRegression(
70     solver='liblinear', multi_class='ovr')))
71 models.append(('LDA', LinearDiscriminantAnalysis()))

```

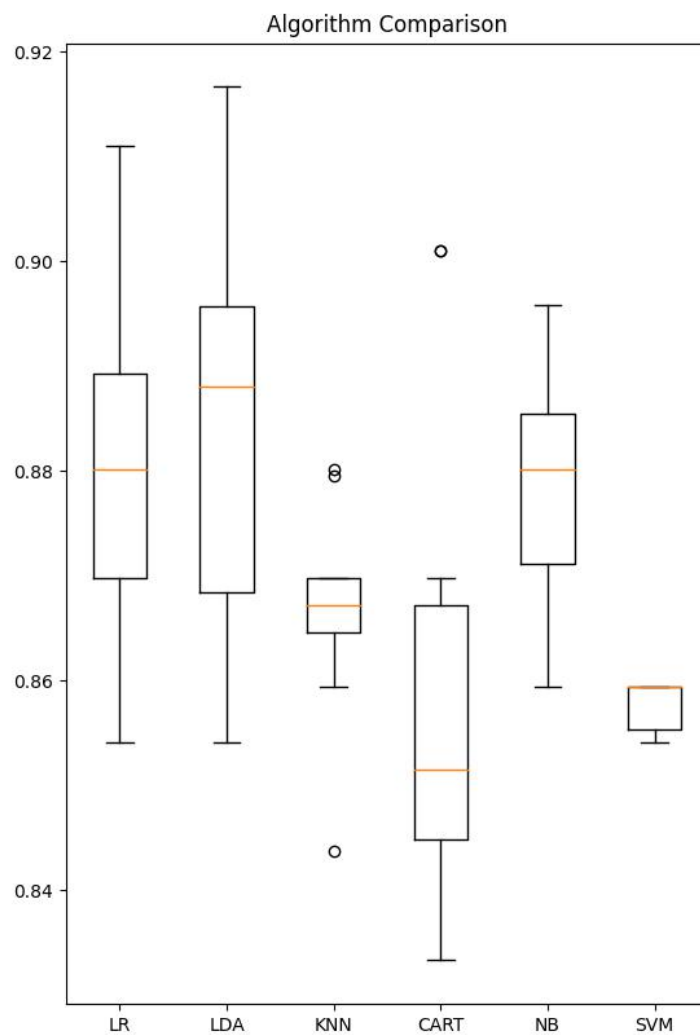
```

1 models.append(('KNN', KNeighborsClassifier()))
2 models.append(('CART', DecisionTreeClassifier()))
3 models.append(('NB', GaussianNB()))
4 models.append(('SVM', SVC(gamma='auto')))
5 results = []
6 names = []
7 for name, model in models:
8     kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
9     cv_results = cross_val_score(
10         model, X_train, Y_train, cv=kfold, scoring='accuracy')
11     results.append(cv_results)
12     names.append(name)
13     print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
14 pyplot.boxplot(results, labels=names)
15 pyplot.title('Algorithm Comparison')
16 pyplot.show()

```

		Бабіч Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пулько І. В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Результати роботи програми:



```
AI/lab2 on ʘ main [?] took 1m 28s 665ms
→ python3 -W ignore LR_2_task4.py
LR: 0.880683 (0.016293)
LDA: 0.885883 (0.019384)
KNN: 0.866604 (0.009853)
CART: 0.860338 (0.022359)
NB: 0.879633 (0.010957)
SVM: 0.857739 (0.002348)
```

Порівнявши алгоритми за показником точності можна дійти висновку, що найкращим є алгоритм NB.

		Бабіч Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пулєко І. В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Код програми:

```
lab2 LR_2_task5.py
1 import seaborn as sns
2 from io import BytesIO # needed for plot
3 import matplotlib.pyplot as plt
4 from sklearn.metrics import confusion_matrix
5 from sklearn import metrics
6 import numpy as np
7 from sklearn.datasets import load_iris
8 from sklearn.linear_model import RidgeClassifier
9 from sklearn.model_selection import train_test_split
10 iris = load_iris()
11 X, y = iris.data, iris.target
12 X_train, X_test, y_train, y_test = train_test_split(
13     X, y, test_size=0.3, random_state=0)
14 clf = RidgeClassifier(tol=1e-2, solver="sag")
15 clf.fit(X_train, y_train)
16 y_pred = clf.predict(X_test)
17 print('Accuracy:', np.round(metrics.accuracy_score(y_test, y_pred), 4))
18 print('Precision:', np.round(metrics.precision_score(
19     y_test, y_pred, average='weighted'), 4))
20 print('Recall:', np.round(metrics.recall_score(
21     y_test, y_pred, average='weighted'), 4))
22 print('F1 Score:', np.round(metrics.f1_score(
23     y_test, y_pred, average='weighted'), 4))
24 print('Cohen Kappa Score:', np.round(
25     metrics.cohen_kappa_score(y_test, y_pred), 4))
26 print('Matthews Corrcoeff:', np.round(
27     metrics.matthews_corrcoef(y_test, y_pred), 4))
28 print('\t\t\tClassification Report:\n',
29     metrics.classification_report(y_pred, y_test))
30 sns.set()
31 mat = confusion_matrix(y_test, y_pred)
32 sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
33 plt.xlabel('true label')
34 plt.ylabel('predicted label')
35 plt.savefig("Confusion.jpg")
36 f = BytesIO()
37 plt.savefig(f, format="svg")
```

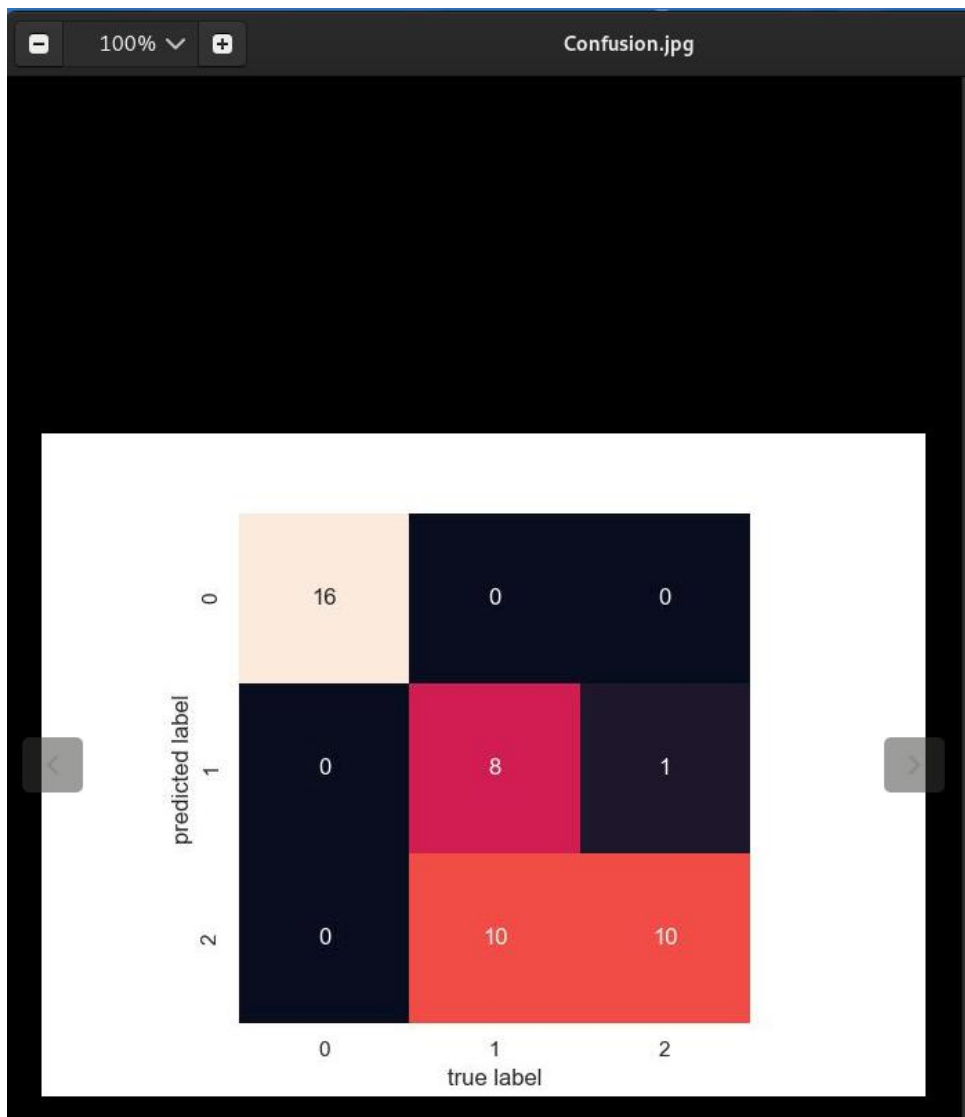
Результат роботи програми:

```
→ python3 -W ignore LR_2_task5.py
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoeff: 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
```

		Бабіч Д. В.			«Житомирська політехніка».23.123.03.000 – Лр2	Арк.
		Пулеко І. В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		



При налаштуванні класифікатора було використано такі параметри: `tol` (точність рішення) та `solver` (рішення для обрахунків). Коефіцієнт Коена Каппа є мірою узгодженості між двома спостерігачами, які оцінюють якісну змінну.

Коефіцієнт кореляції Метьюза використовується для оцінки зв'язку між результатами діагностики та станом пацієнта. Він розраховується як відношення кількості правильних діагнозів до загальної кількості діагнозів, враховуючи відсутність зв'язку випадкових результатів.

Висновки: у ході виконання даної лабораторної роботи я, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив класифікацію даних різними методами та за допомогою метрик якості класифікації оцінив їх ефективності.