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)

| Назва | Опис | Тип значень |
|---|---|---|
| age | Вік | Числове |
| workclass | Вид парцевлаштування | Категоріальне |
| fnlwgt | Кількість осіб, які мають такі ж ознаки | Числове |
| education | Освіта | Категоріальне |
| education-num | Років навчання | Числове |
| marital-status | Сімейне положення | Категоріальне |
| occupation | Професія | Категоріальне |
| relationship | Відносини | Категоріальне |
| race | Раса | Категоріальне |
| sex | Стать | Категоріальне |
| capital-gain | Приріст капіталу | Числове |
| capital-loss | Втрата капіталу | Числове |
| hours-per-week | Кількість робочих годин на тиждень | Числове |
| native-country | Країна походження | Категоріальне |

Accuracy score: 62.64%

Precision score: 69.18%

Recall score: 38.24%

F1 score: 56.15%

Тестова точка - <=50K. Отже тестова точка має дохід менше 50 тисяч в рік.

**Код програми:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дата* | | | |
| *Розроб.* | | *Горбенко Д.С.* | | | Звіт з лабораторної роботи №2 | *Літ.* | *Арк.* 1 / *Аркушів* |
| *Перевір.* | | *Пулеко І.В..* | | | | | |
| *Керівник* | | | | | | *ФІКТ Гр. ПІ-59(І)* | |
| *Н. контр.* | | | | | | | |
| *Затверд.* | | | | | | | |

```
import numpy as np
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

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

# Читання даних
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(', ')
        income_class = data[-1]
        if income_class == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

        if income_class == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

# Перетворення на масив numpy
X = np.array(X)

# Перетворення рядкових даних на числові
label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        current_label_encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

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

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

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

precision = cross_val_score(classifier, X, y, scoring='precision', cv=3)
print("Precision score: " + str(round(100 * precision.mean(), 2)) + "%")
```

```
recall = cross_val_score(classifier, X, y, scoring='recall', cv=3)
print("Recall score: " + str(round(100 * recall.mean(), 2)) + "%")

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Han-
dlers-cleaners', 'Not-in-family', 'White',
              'Male',
              '0', '0', '40', 'United-States']

# Кодування тестової точки даних
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:
        encoder = label_encoder[count]
        input_data_encoded[i] = int(encoder.transform([(input_data[i])])[-1])
        count += 1

input_data_encoded = np.array(input_data_encoded)

# Використання класифікатора для кодованої точки даних
# та виведення результату
predicted_class = classifier.predict([input_data_encoded])
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

**Завдання 2.2.** Порівняння якості класифікаторів SVM з нелінійними ядрами

**З поліноміальним ядром:**

Accuracy score: 58.41%

Precision score: 41.6%

Recall score: 33.05%

F1 score: 46.5%

**З гаусовим ядром:**

Accuracy score: 78.61%

Precision score: 98.72%

Recall score: 14.26%

F1 score: 71.95%

**З сигмоїдальним ядром:**

Accuracy score: 63.89%

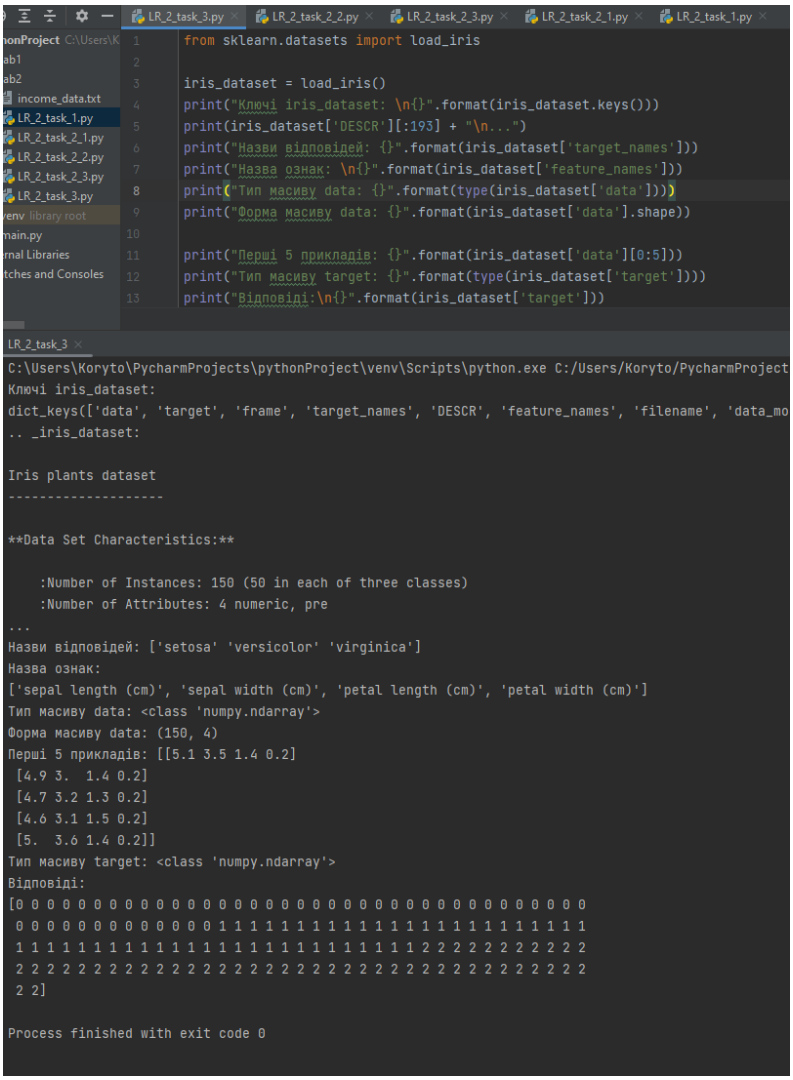
Precision score: 27.01%

Recall score: 26.48%

F1 score:63.77%

Найбільш точним виявився CVM класифікатор з гаусовим ядром.

**Завдання 2.3.** Порівняння якості класифікаторів на прикладі класифікації сортів ірисів



```
from sklearn.datasets import load_iris

iris_dataset = load_iris()
print("Ключі iris_dataset: \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("Перші 5 прикладів: {}".format(iris_dataset['data'][0:5]))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

```
C:\Users\Koryto\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Koryto/PycharmProject
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_mo
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Перші 5 прикладів: [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
Тип масиву target: <class 'numpy.ndarray'>
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

Process finished with exit code 0
```

рис.1 Код для ознайомлення зі структурою даних та результати

Графіки

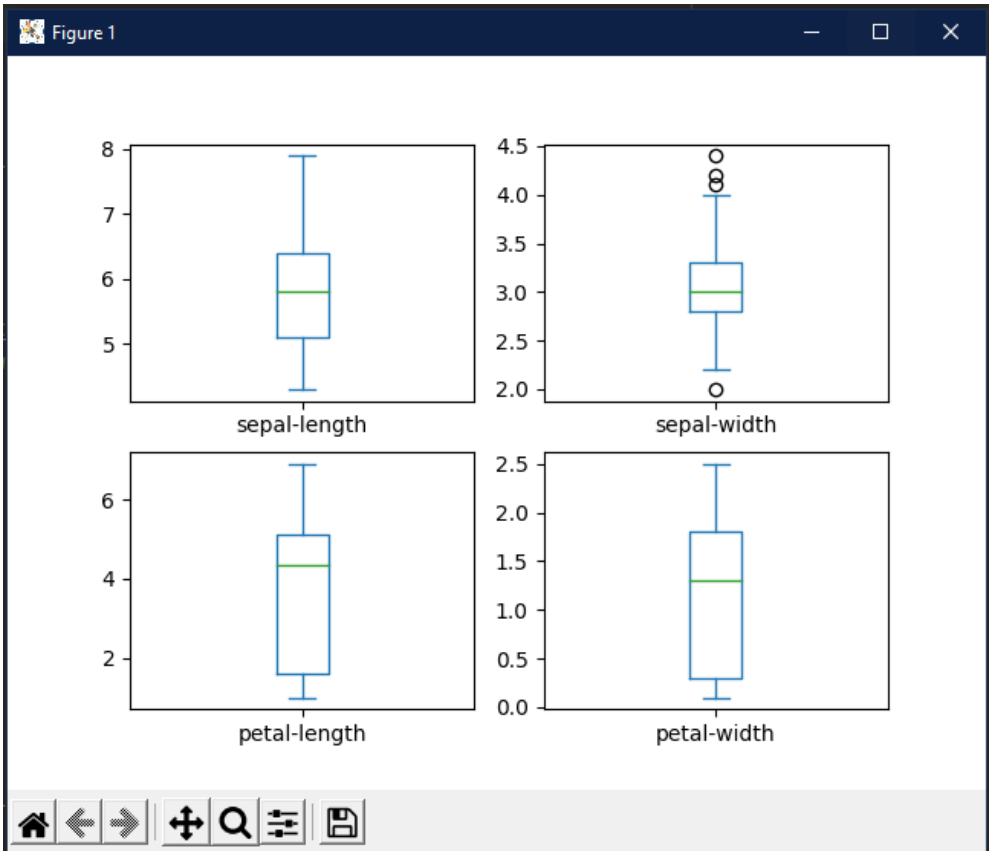



рис.2 Одновимірні графіки характеристик

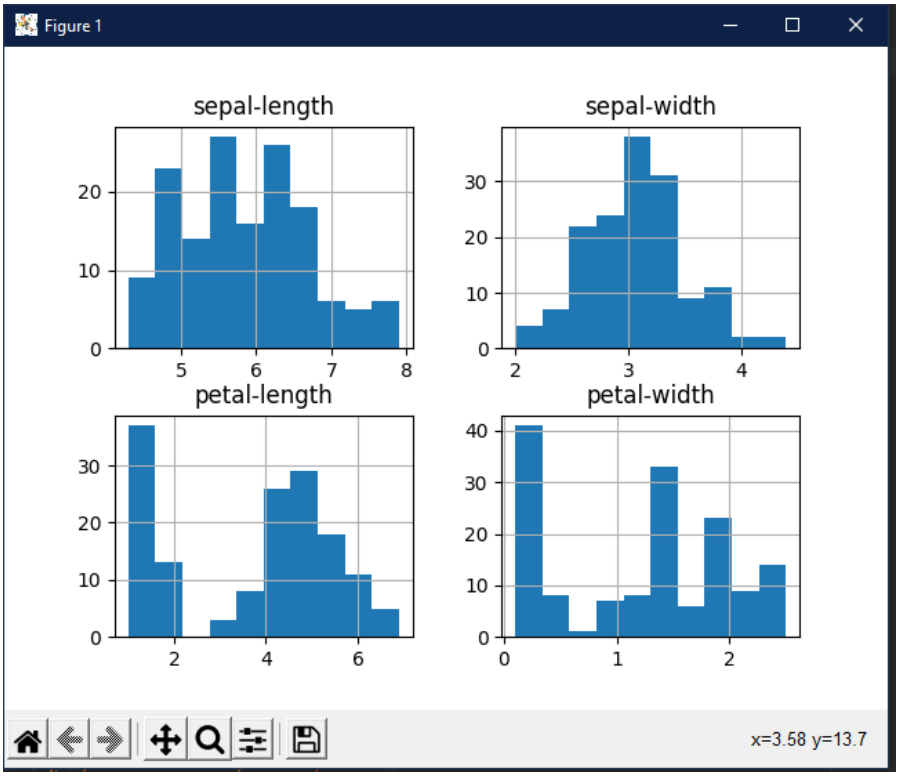



рис.3 Діаграма розмаху атрибутів

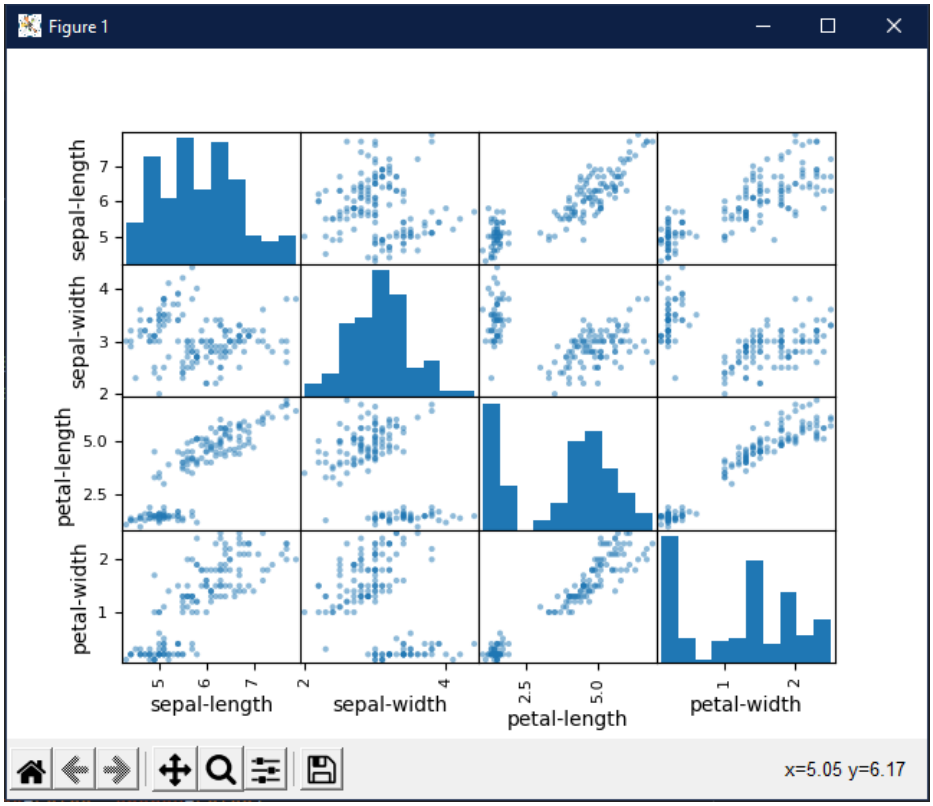



рис.4 Матриця розсіювання

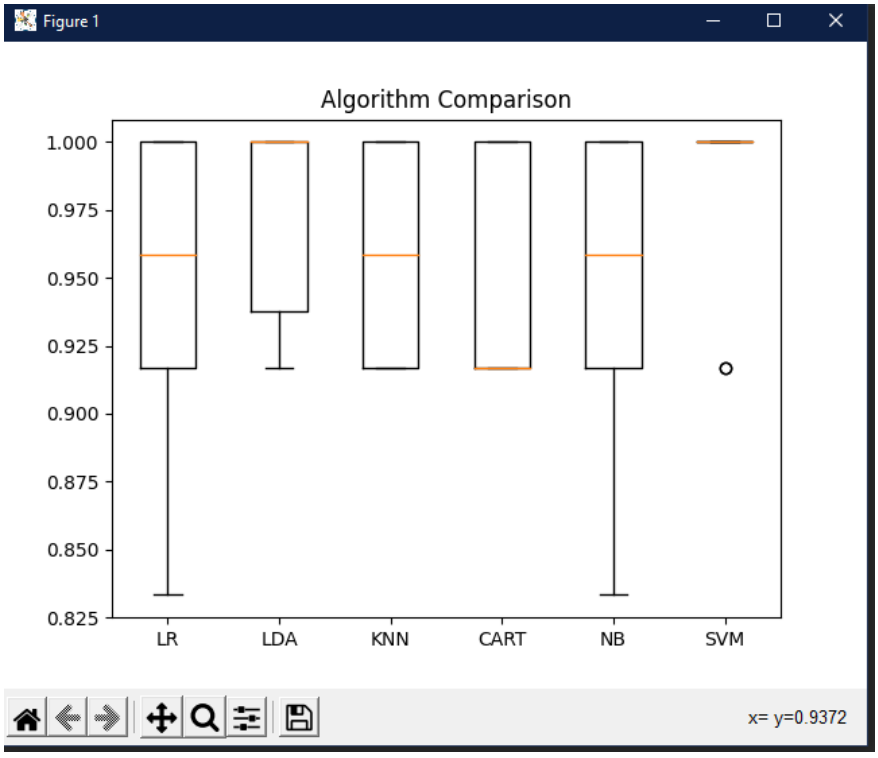



рис.5 Графік порівняння алгоритмів

Проаналізувавши ортиманий графік, я обрав метод класифікації CVM, тому що він показав найвищу якість.

Код програми:

```
from sklearn.datasets import load_iris
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
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
import numpy as np

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)

# shape
print(dataset.shape)
# Зріз даних head
print(dataset.head(20))
# Стастичні зведення методом describe
print(dataset.describe())
# Розподіл за атрибутом class
print(dataset.groupby('class').size())

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

# Гістограма розподілу атрибутів датасета
dataset.hist()
pyplot.show()

# Матриця діаграм розсіювання
scatter_matrix(dataset)
pyplot.show()

# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
# Вибір перших 4-х стовпців
X = array[:,0:4]
# Вибір 5-го стовпця
y = array[:,4]
# Разделение X и y на обучающую и контрольную выборки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)

# Завантажуємо алгоритми моделі
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')))

# оцінюємо модель на кожній ітерації
```

```
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='accura-
cy')
  results.append(cv_results)
  names.append(name)
  print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

# Порівняння алгоритмів
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()

# Створюємо прогноз на контрольній вибірці
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# Оцінюємо прогноз
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масиву X_new: {}".format(X_new.shape))

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

```
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)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.040825)
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

Форма масиву X_new: (1, 4)
Прогноз: ['Iris-setosa']
Спрогнозована мітка: Iris-setosa

Process finished with exit code 0
```

рис.5 Результат виконання

**Завдання 2.4.** Порівняння якості класифікаторів для набору даних завдання 2.1

Код програми:

```
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
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
import numpy as np

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

# Читання даних
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(', ')
    income_class = data[-1]
    if income_class == '<=50K' and count_class1 < max_datapoints:
      X.append(data);
      count_class1 += 1

    if income_class == '>50K' and count_class2 < max_datapoints:
      X.append(data);
      count_class2 += 1

# Перетворення на масив numpy
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:
    current_label_encoder = preprocessing.LabelEncoder()
    label_encoder.append(current_label_encoder)
    X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

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

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

# Разделение X и y на обучающую и контрольную выборки
X_train, X_test, Y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)

# Завантажуємо алгоритми моделі
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')))

# оцінюємо модель на кожній ітерації
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='accura-
cy')
  results.append(cv_results)
  names.append(name)
  print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

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

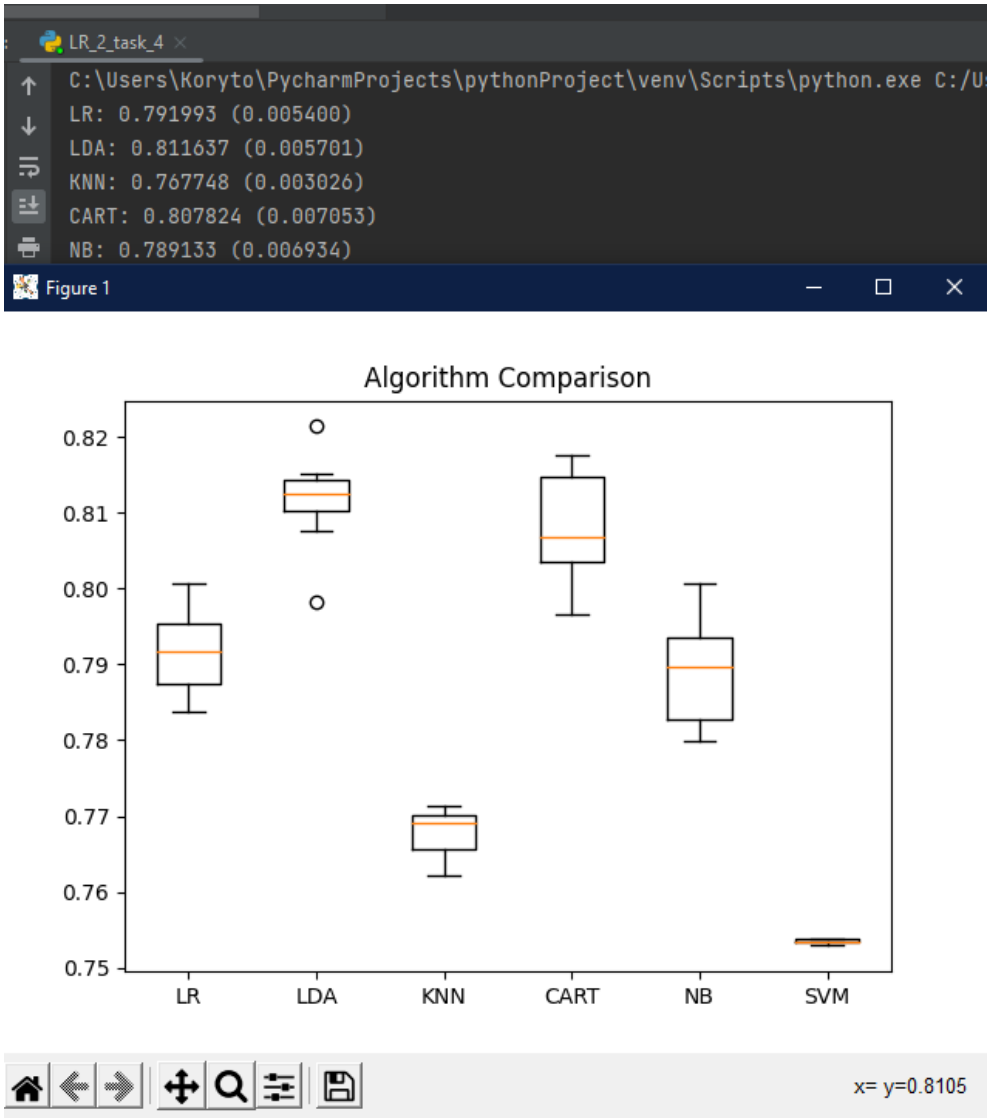



рис.6 Результат виконання

**Завдання 2.5.** Класифікація даних лінійним класифікатором Ridge

Код програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from io import BytesIO #neded for plot
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt


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, ran-
dom_state = 0)
clf = RidgeClassifier(tol = 1e-2, solver = "sag")
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

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

mat = confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square = True, annot = True, fmt = 'd', cbar = False)
plt.xlabel('true label')
plt.ylabel('predicted label');
plt.savefig("Confusion.jpg")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format = "svg")
```

```
C:\Users\Koryto\PycharmProjects\pythonProject\venv\Scripts\python.ex
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
```

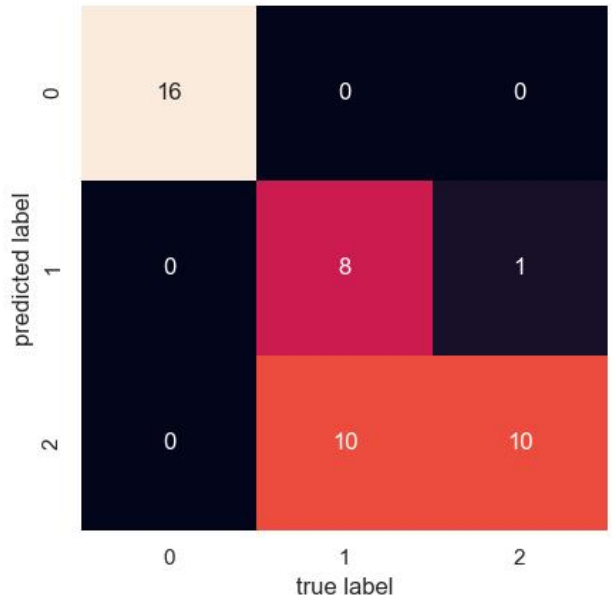



рис.7 Результат виконання

Класифікатор має наступні параметри:

- tol – точність класифікації
- solver – алгоритм, який виконує класифікацію

На зображені Confusion.jpg наведені результати класифікації. На вертикальній шкалі відкладені наявні класи ірису в числовій репрезентації, а на горизонтальній передбаченя класи ірису. Цифра на перетині – кількість результатів системи при справжньому і передбаченому класі.

Коефіцієнт кореляції Метьюза – коефіцієнт, який на основі матриці помилок вираховує коефіцієнт від -1 до 1, де 1 – є результатом ідеальної класифікації, а 0 – рівень випадкового вибору.

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Коефіцієнт Коена Каппа – коефіцієнт, який також за основу бере матрицю помилок, але замість загальної якості, звертає увагу на нерівноцінне розподілення класів.

**Висновки**: на даній лабораторній роботі я дослідив різні методи класифікації даних та навчився їх порівнювати, використовуючи спеціалізовані бібліотеки та мову програмування Python.