ЛАБОРАТОРНА РОБОТА № 2
ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Мета роботи: *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)

Опис ознак

Значення	Назва	Опис	Вид
bachelors	education	освіта	категорія
13	education-num	найвищий рівень освіти	число
state-gov	workclass	тип зайнятості	категорія
39	age	вік	число
never-married	marital-status	сімейний стан	категорія
adm-clerical	occupation	професія	категорія
male	sex	стать	категорія
white	race	раса	категорія
not-in-family	relationship	відносини	відносини
77516	fnlwgt	final weight	число
united-states	native-country	країна	категорія
2174	capital-gain	капітал	число
0	capital-loss	збитки	число
40	hours-per-week	кількість годин роботи за тиждень	число
<=50k	label	мітка заробітку більше 50 тисяч	число

```
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
classifier = OneVsOneClassifier(LinearSVC(random_state=0, max_iter=1000))
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=5)
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
```

Рис. 1. Неповне відображення коду програми.

Отже за результатами на рис.2. людина буде мати заробітню плату більше 50 тисяч доларів.

```
Accuracy: 81.95%
Precision: 80.94%
Recall: 81.95%
F1: 80.13%
F1 score: 80.13%
>50K

Process finished with exit code 0
```

Рис. 2. Результат показників якості класифікації та результат прогнозу.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

```python
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
classifier = SVC(kernel='poly', degree=6)
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
```

Рис. 3. Неповний код програми з використанням поліноміального ядра.

```
/usr/bin/python3.10 /home/xtr99/labs/ai/lab02/LR_2_task_2_1.py
Accuracy: 84.22%
Precision: 83.47%
Recall: 84.22%
F1: 83.35%
F1 score: 83.35%
<=50K
```

Рис. 4. Результат виконання програми з використанням поліноміального ядра.

LR_2_task_2_2.py

```
            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)
    scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
    X = scaller.fit_transform(X)
    classifier = SVC(kernel='rbf')
    classifier.fit(X=X, y=Y)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=5)
    scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
    X_train = scaller.fit_transform(X_train)
    classifier.fit(X=X_train, y=y_train)
    y_test_pred = classifier.predict(X_test)
    f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
    accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
```

Рис. 5. Неповний код програми з використанням гаусове ядро.

```
/usr/bin/python3.10 /home/xtr99/labs/ai/lab02/LR_2_task_2_2.py
Accuracy: 83.96%
Precision: 83.18%
Recall: 83.96%
F1: 82.95%
F1 score: 82.95%
<=50K
```

Рис. 6. Результат виконання програми з гаусова ядра.

```
Y = X_encoded[:, -1].astype(int)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
classifier = SVC(kernel='sigmoid')
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
```

Рис. 7. Неповний код програми з використанням сигмоїдальне ядро.

```
/usr/bin/python3.10 /home/xtr99/labs/ai/lab02/LR_2_task_2_3.py
Accuracy: 57.26%
Precision: 57.1%
Recall: 57.26%
F1: 57.18%
F1 score: 57.18%
<=50K
```

Рис. 8. Неповний код програми з використанням сигмоїдальне ядро.

Отже за результатами найкраще використання з Гаусовим ядром, насправді мають добру точність та швидкість виконання. Поліноміальне ядро має гарну точність але довгий процес виконання, а сигмоїдальне ядро з поганою точністю та процесом з тривалим виконанням.

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

LR_2_task_3.py

```
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("Назви ознак: \n{}".format(iris_dataset["feature_names"]))
print("Тип масиву date: {}".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{}".format(iris_dataset['target']))
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))
# Статичні 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
```

Рис. 9. Неповний код програми.

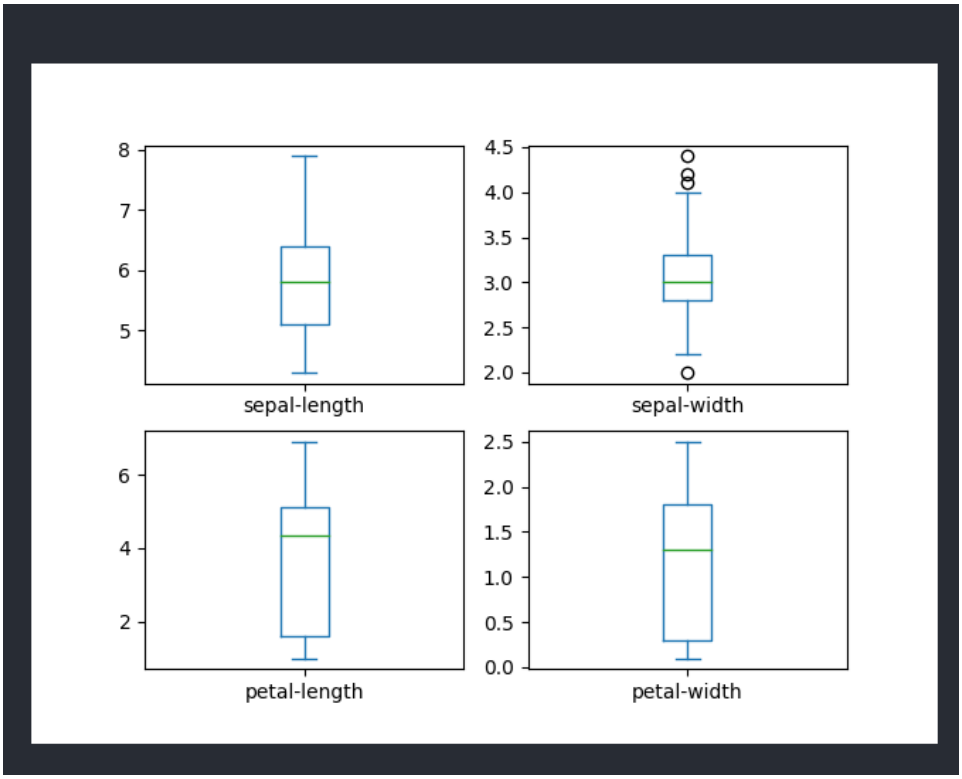


Рис. 10. Діаграма розмаху.

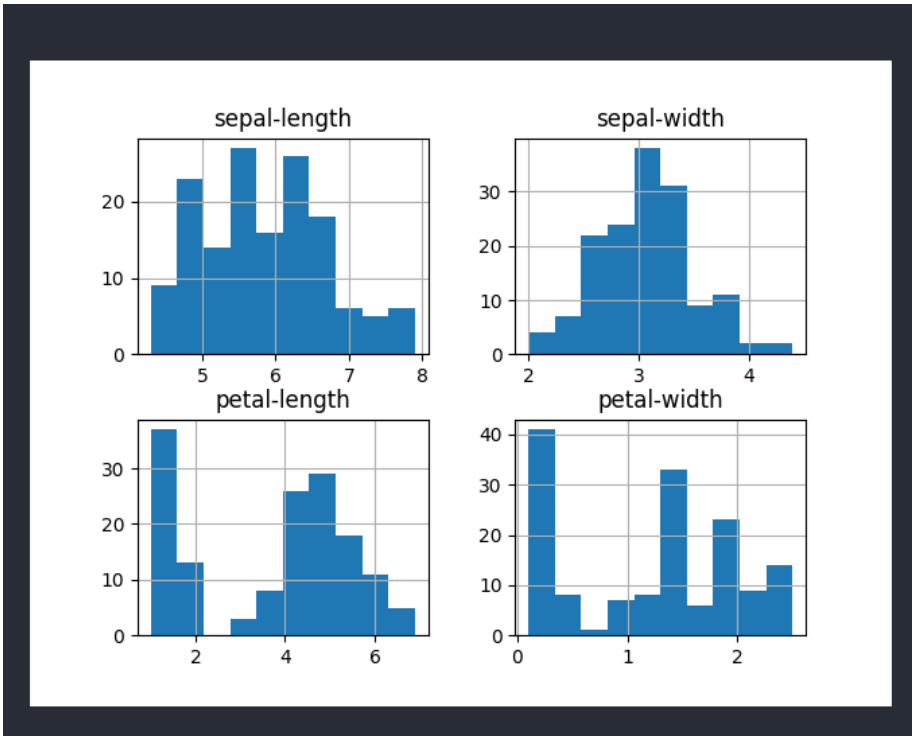


Рис. 11. Діаграми розподілу атрибутів.

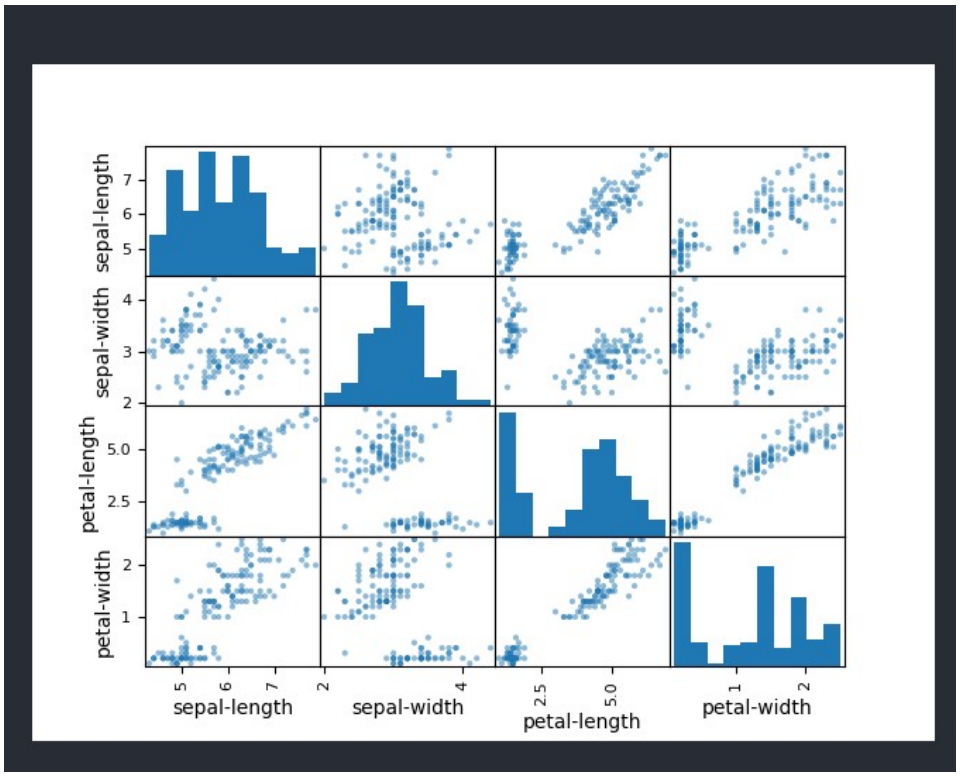


Рис. 12. Діаграми розсіювання.

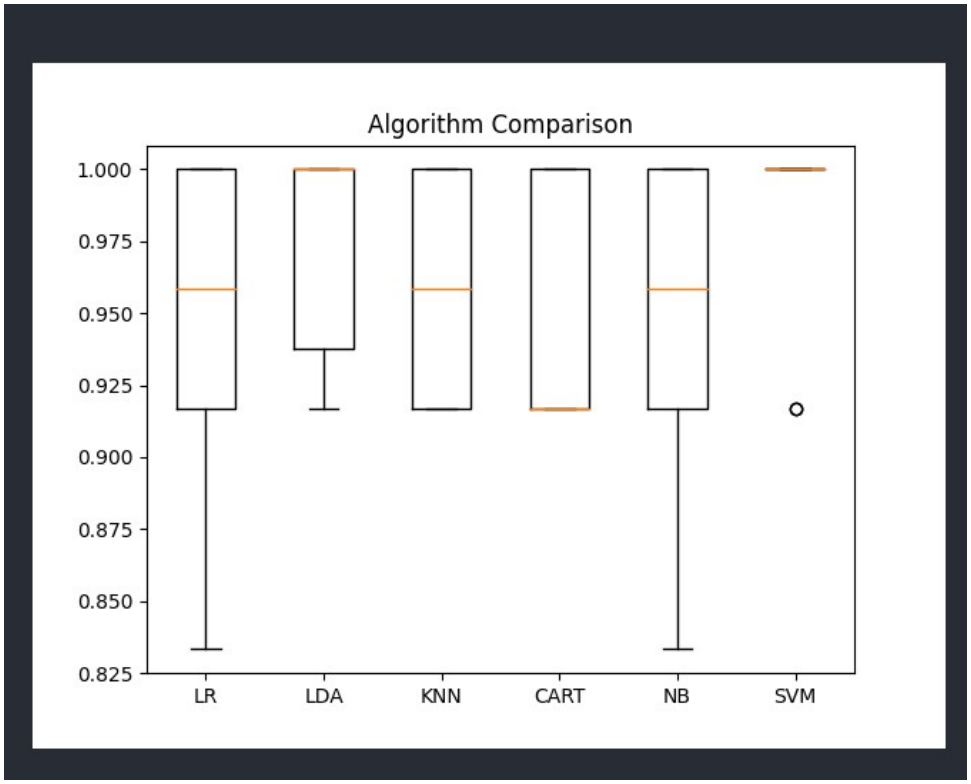
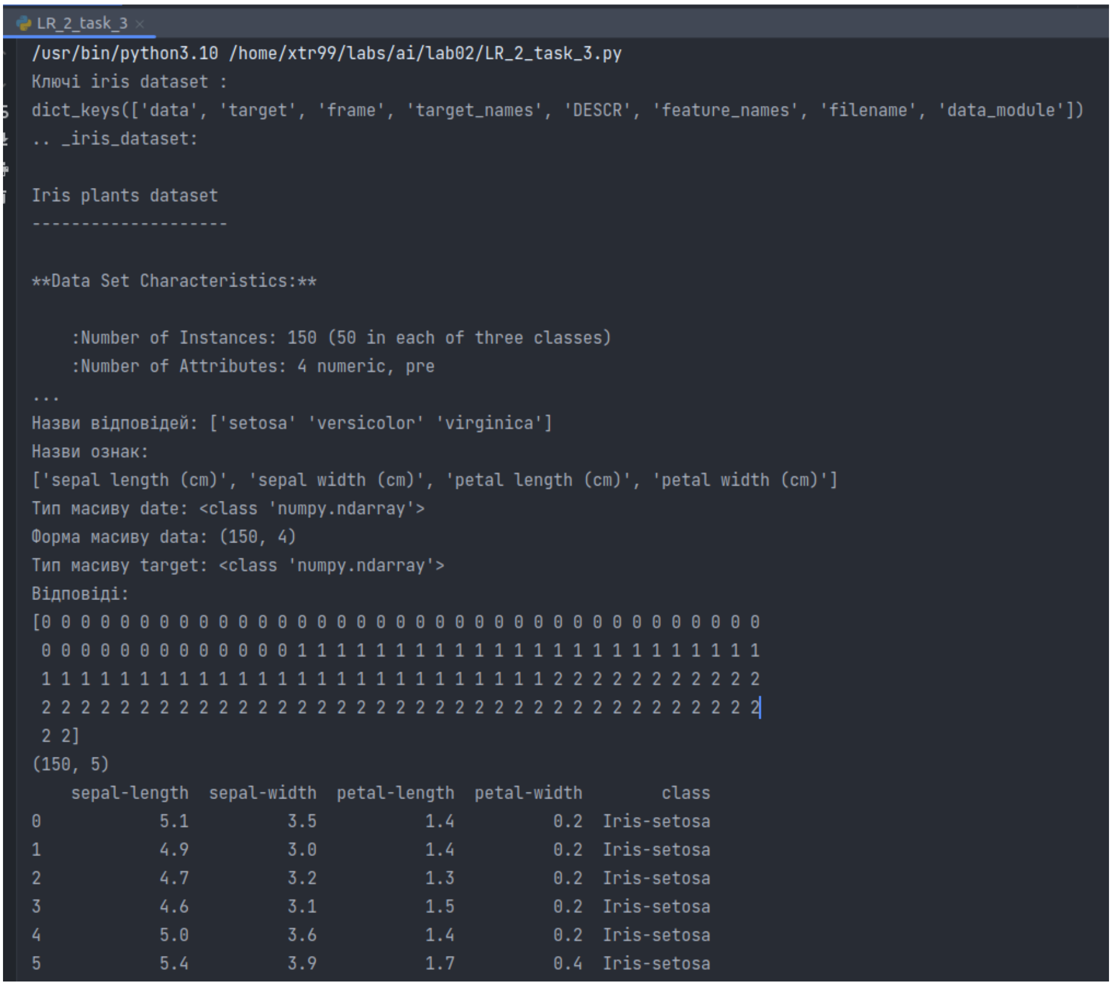


Рис. 13. Графік порівняння алгоритмів.



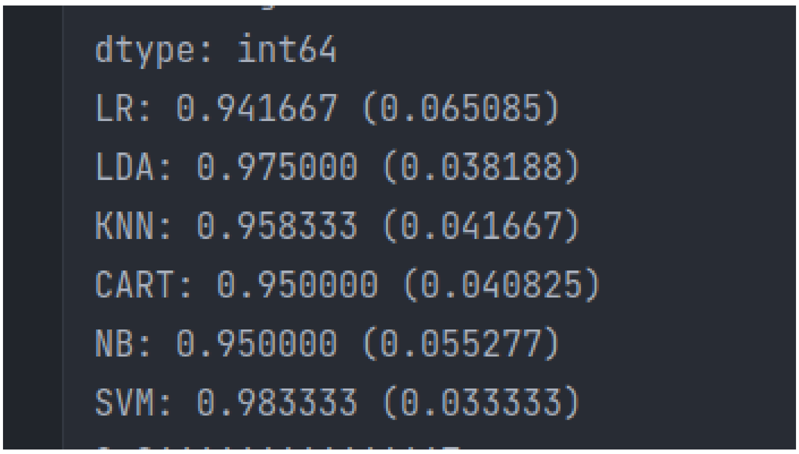
```
/usr/bin/python3.10 /home/xtr99/labs/ai/lab02/LR_2_task_3.py
Ключі iris dataset :
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назви ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву date: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
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 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]
(150, 5)
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
```

Рис. 14. Результат виконання програми.



```
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)
```

Рис. 15. Результат оцінки моделей.

```
Прогноз: ['Iris-setosa']
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
```

Рис. 16. Результат показників якості та прогноз.

Модель SVM (Support Vector Machine) має найвищу показник точності, що становить 0.983333, зі стандартним відхиленням 0.033333. Це означає, що SVM правильно класифікує 98.33% випадків.Тому, на основі результатів можна зробити висновок, що модель SVM є найкращою з точки зору точності класифікації. Вона має високу точність і стабільність, що робить її привабливим вибором для багатьох задач класифікації.Квітка з кроку 8 належить до класу Iris-setosa.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

```
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=0.20, random_state=1)
models = [('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
          ('LDA', LinearDiscriminantAnalysis()),
          ('KNN', KNeighborsClassifier()),
          ('CART', DecisionTreeClassifier()),
          ('NB', GaussianNB()),
          ('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()
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print('SVC')
print(classification_report(Y_validation, predictions))
model = GaussianNB()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print('Gaussian')
print(classification_report(Y_validation, predictions))
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

Рис. 17. Неповний код програми.

```
LR: 0.818849 (0.004427)
LDA: 0.812176 (0.003802)
KNN: 0.817606 (0.003760)
CART: 0.804509 (0.006445)
NB: 0.799080 (0.005377)
SVM: 0.824112 (0.005380)
```

Рис. 18. Результат оцінки моделей.

Метод опорних векторів

	precision	recall	f1-score	support
0	0.84	0.95	0.89	4483
1	0.78	0.45	0.57	1550
accuracy			0.83	6033
macro avg	0.81	0.70	0.73	6033
weighted avg	0.82	0.83	0.81	6033

Рис. 19. Результат показників якості для SVM.

Наївний баєсовський класифікатор

	precision	recall	f1-score	support
0	0.81	0.95	0.87	4483
1	0.71	0.35	0.47	1550
accuracy			0.80	6033
macro avg	0.76	0.65	0.67	6033
weighted avg	0.78	0.80	0.77	6033

Рис. 20. Результат показників якості для NB.

Класифікація та регресія за допомогою дерев

	precision	recall	f1-score	support
0	0.87	0.87	0.87	4483
1	0.61	0.61	0.61	1550
accuracy			0.80	6033
macro avg	0.74	0.74	0.74	6033
weighted avg	0.80	0.80	0.80	6033

Рис. 21. Результат показників якості для CART.

Метод k-найближчих сусіді

	precision	recall	f1-score	support
0	0.86	0.91	0.88	4483
1	0.68	0.57	0.62	1550
accuracy			0.82	6033
macro avg	0.77	0.74	0.75	6033
weighted avg	0.81	0.82	0.82	6033

Рис. 22. Результат показників якості для KNN.

Лінійний дискримінантний аналіз

	precision	recall	f1-score	support
0	0.82	0.94	0.88	4483
1	0.71	0.42	0.52	1550
accuracy			0.81	6033
macro avg	0.76	0.68	0.70	6033
weighted avg	0.79	0.81	0.79	6033

Рис. 23. Результат показників якості для LDA.

Логістична регресія

	precision	recall	f1-score	support
0	0.83	0.94	0.88	4483
1	0.72	0.45	0.56	1550
accuracy			0.81	6033
macro avg	0.78	0.70	0.72	6033
weighted avg	0.80	0.81	0.80	6033

Рис. 24. Результат показників якості для LR.

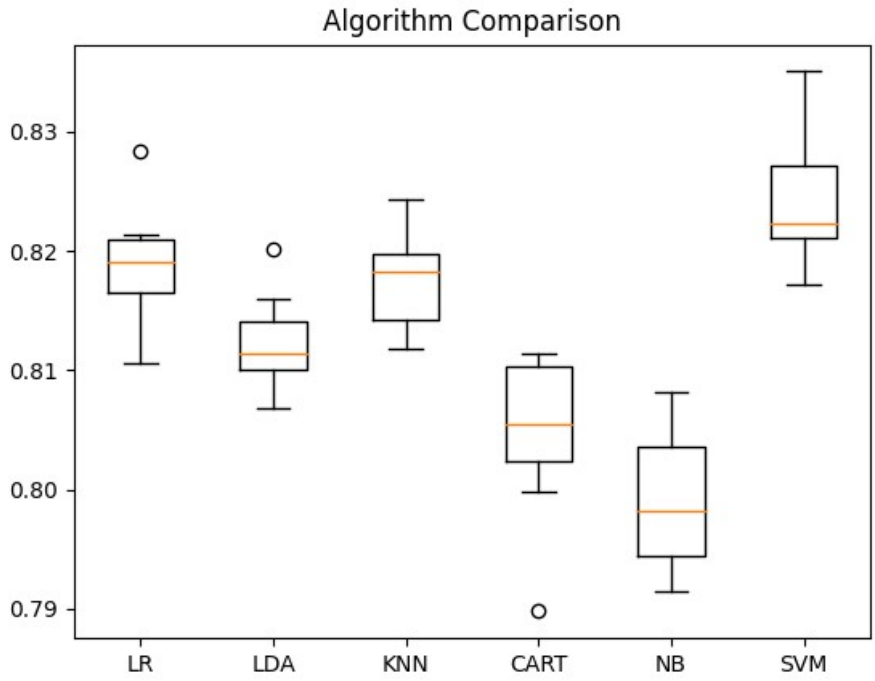


Рис. 25. Графік порівняння алгоритмів.

За основними показниками точності, модель SVM демонструє найвищу точність з результатом 0.83. Це означає, що SVM класифікує дані з високою точністю, і має низьку варіабельність в результаті. SVM часто є потужним інструментом для класифікації, оскільки воно може побудувати оптимальний гіперплощину для розділення класів, особливо в задачах з нелінійними залежностями. В даних результатів, SVM показує високу точність та стабільність, що свідчить про його ефективність у цій конкретній задачі класифікації.Враховуючи це, модель SVM може бути вважається найкращою для цієї задачі класифікації, оскільки вона досягає найвищих показників точності та має найнижчу варіабельність у порівнянні з іншими моделями.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge.

```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from io import BytesIO
import matplotlib.pyplot as plt
from sklearn import metrics
sns.set()
iris = load_iris()
X, y = iris.data, iris.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3, random_state=0)
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)
ypred = clf.predict(Xtest)
print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))
print('Precision:', np.round(metrics.precision_score(ytest, ypred, average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(ytest, ypred, average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(ytest, ypred, average='weighted'), 4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(ytest, ypred), 4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n',metrics.classification_report(ypred, ytest))
mat = confusion_matrix(ytest, 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")
```

Рис. 26.Код програми.

```
/usr/bin/python3.10 /home/xtr99/labs/ai/lab02/LR_2_task_5.py
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
```

Рис. 27. Результат виконання.

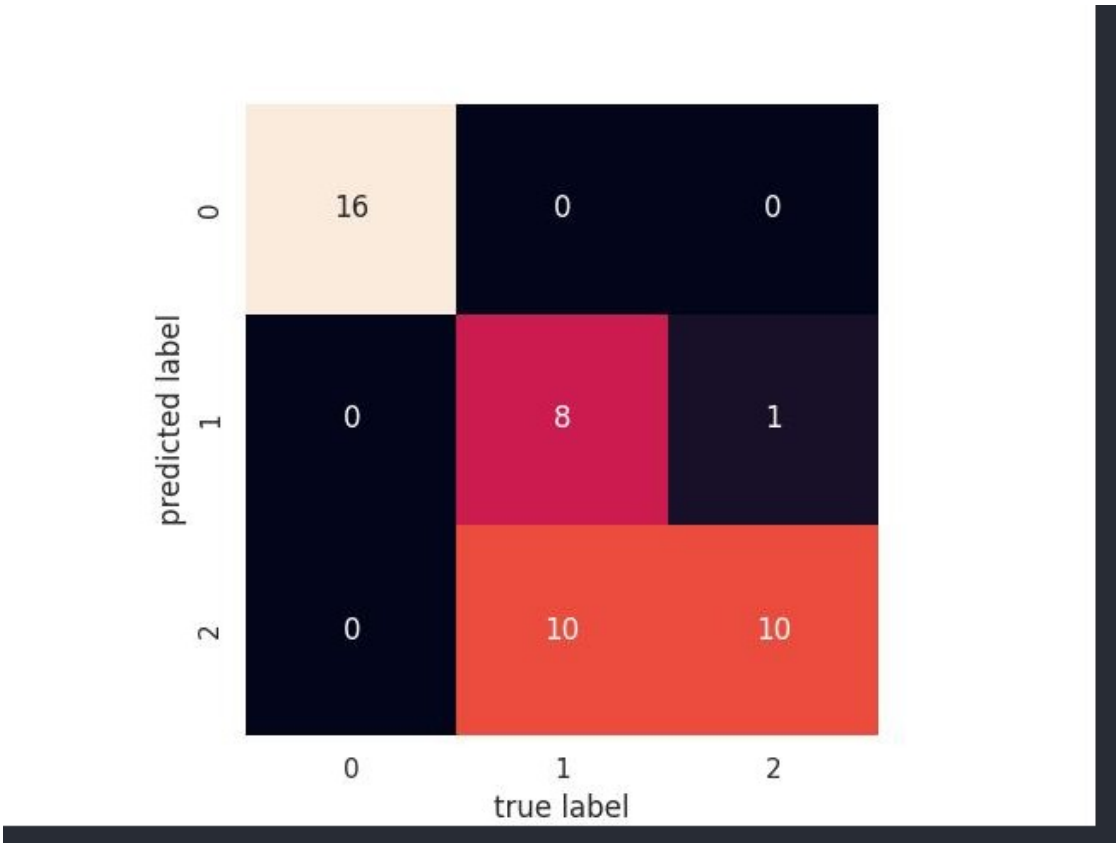


Рис. 28. Матриця невідповідності.

Матриця невідповідності є спеціальною таблицею, яка дозволяє візуалізувати ефективність алгоритму. Кожен рядок матриці представляє прогнозовані класи, тоді як кожен стовпець представляє справжні класи.

Коефіцієнт Коена - це статистика, яка використовується для оцінки надійності між експертами у якісних аспектах.

Кореляція Метьюза - це метрика, яка використовується в машинному навчанні для вимірювання якості бінарних або мультикласових класифікацій. Вона враховує частину правильно класифікованих екземплярів, а також частину неправильно класифікованих екземплярів, що дозволяє оцінити якість класифікаційної моделі.

Покликання на github: https://github.com/mrkulish/ai-labs/tree/master/lab02

Висновок: при виконанні лабораторної роботи навчилися використовувати спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчилися їх порівнювати.