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). Створіть класифікатор у вигляді машини опорних векторів, призначений для прогнозування меж доходу заданої фізичної особи на основі 14 ознак (атрибутів). Метою є з'ясування умов, за яких щорічний прибуток людини перевищує $50000 або менше цієї величини за допомогою бінарної класифікації.

Таблиця 1

Опис ознак на прикладі першого запису

Значення	Назва	Опис	Вид
39	age	вік особи	число
State-gov	workclass	тип зайнятості (не працює, не оплачено, самозайнятий тощо)	категорія
77516	fnlwgt	final weight – кількість таких людей	число
Bachelors	education	степінь освіти	категорія
13	education-num	найвищий рівень освіти, який має особа	число
Never-married	marital-status	сімейний стан	категорія
Adm-clerical	occupation	рід зайнятості, професія	категорія
Not-in-family	relationship	відносини людини	категорія
White	race	раса	категорія
Male	sex	стать	категорія
2174	capital-gain	приріст капіталу особи	число
0	capital-loss	збитки	число
40	hours-per-week	скільки годин особа працює на тиждень	число
United-States	native-country	країна походження	категорія
<=50K	label	мітка чи заробляє особа більше 50 тисяч	число

```python
import numpy as np
from sklearn import preprocessing, svm
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 = 'Task/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(', ')

        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

# 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:
        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)
```

Рис. 1. Код програми

```python
# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))

# Навчання класифікатора
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, max_iter=1000))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# Обчислення F-міри для SVM-класифікатора
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', 'Handlers-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:
        input_data_encoded[i] = int(label_encoder[count].transform([input_data[i]]))
        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.reshape(1, 14))
print("Label", label_encoder[-1].inverse_transform(predicted_class)[0])

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. 2. Код програми

```
F1 score: 56.15%
Accuracy: 62.64%
Precision: 75.88%
Recall: 62.64%
```

Рис. 3. Показники якості класифікації

```
Label <=50K
```

Рис. 4. Передбачений результат

За результатами виконання коду можна зробити висновок, що особа з заданими вхідними парамерами, ймовірно, буде мати заробітну платню менше за 50 тисяч доларів.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

```
from sklearn import svm
from sklearn.model_selection import train_test_split
from methods import get_X_y, get_result, show_quality_values

X, y, label_encoder = get_X_y()

# Створення SVM-класифікатора
classifier = svm.SVC(kernel='poly', degree=8, max_iter=50000, 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 = svm.SVC(kernel='poly', degree=8, max_iter=50000, random_state=0)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

label = get_result(classifier, label_encoder)
print("Label", label)

show_quality_values(classifier, X, y)
```

Рис. 5. Код програми для поліноміального ядра

```
from sklearn import svm
from sklearn.model_selection import train_test_split
from methods import get_X_y, get_result, show_quality_values

X, y, label_encoder = get_X_y()

# Створення SVM-класифікатора
classifier = svm.SVC(kernel='rbf', max_iter=50000, 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 = svm.SVC(kernel='rbf', max_iter=50000, random_state=0)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

label = get_result(classifier, label_encoder)
print("Label", label)

show_quality_values(classifier, X, y)
```

Рис. 6. Код програми для гаусового ядра

```python
from sklearn import svm
from sklearn.model_selection import train_test_split
from methods import get_X_y, get_result, show_quality_values

X, y, label_encoder = get_X_y()

# Створення SVM-класифікатора
classifier = svm.SVC(kernel='sigmoid', max_iter=50000, 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 = svm.SVC(kernel='sigmoid', max_iter=50000, random_state=0)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

label = get_result(classifier, label_encoder)
print("Label", label)

show_quality_values(classifier, X, y)
```

Рис. 7. Код програми для сигмоїдального ядра

```
Label <=50K
F1 score: 69.64%
Accuracy: 76.21%
Precision: 73.09%
Recall: 76.21%
```

Рис. 8. Результат виконання програми для поліноміального ядра

```
Label <=50K
F1 score: 69.73%
Accuracy: 76.24%
Precision: 73.17%
Recall: 76.24%
```

Рис. 9. Результат виконання програми для гаусового ядра

```
Label <=50K
F1 score: 66.74%
Accuracy: 74.34%
Precision: 67.07%
Recall: 74.34%
```

Рис. 10. Результат виконання програми для сигмоїдального ядра

Таблиця 2

Порівняння показників якості

Ядро	F-міра	Акуратність	Точність	Повнота
Лінійне	56,15%	62,64%	62,64%	75,88%
Поліноміальне	69,64%	76,21%	73,09%	76,21%
Гаусове	69,73%	76,24%	73,17%	76,24%
Сигмоїдальне	66,74%	74,34%	67,07%	74,34%

При порівнянні результатів можна зробити висновок, що гаусову ядро має найвищі показники.

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(iris_dataset['DESCR'][193:-1])
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))

# - Виведіть значення ознак для перших п'яти прикладів
print("Перші 5 прикладів: \n{}".format(iris_dataset['data'][0:5]))

print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

Рис. 11. Код програми для ознайомлення з даними

```
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']
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 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]
```

Рис. 12. Результат виконання коду

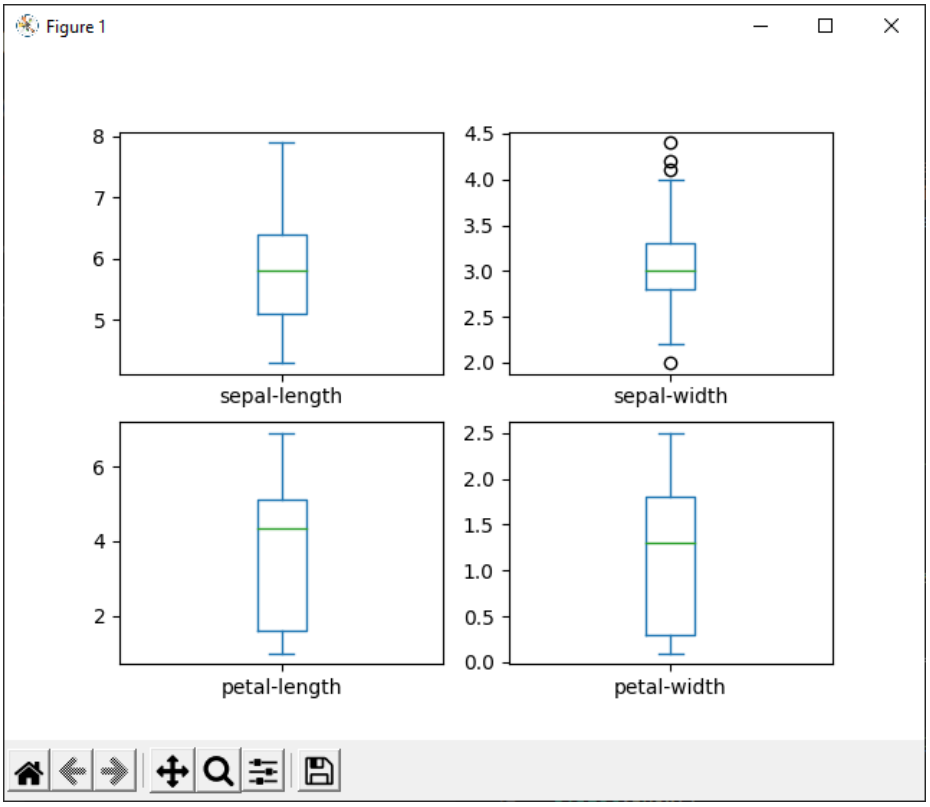

Рис. 13. Діаграма розмаху

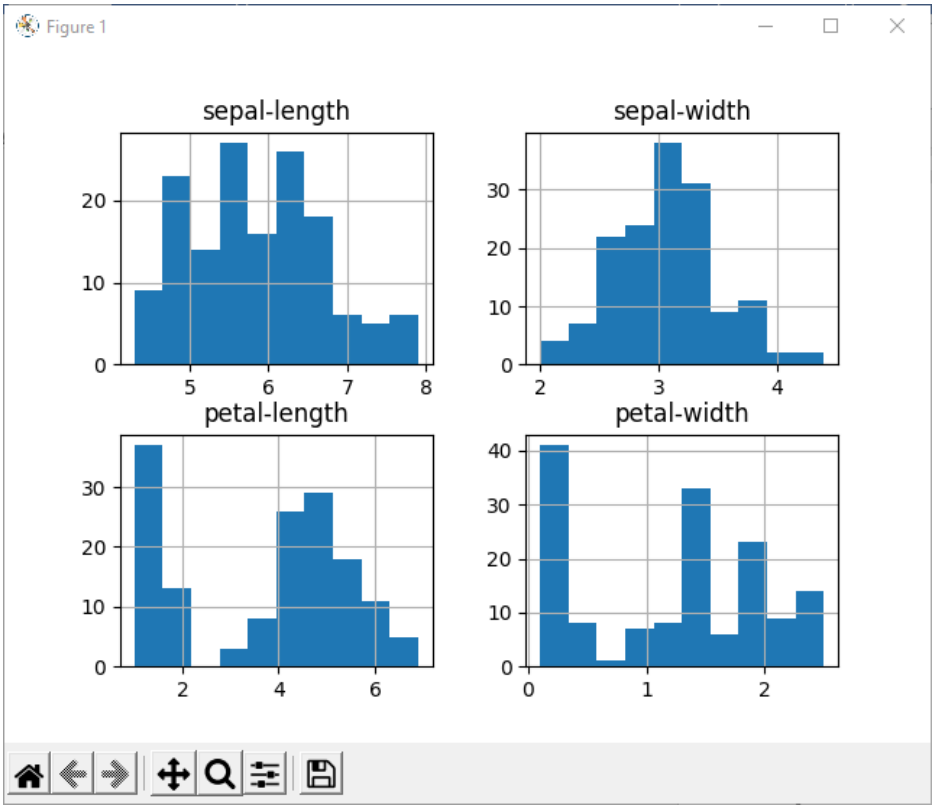

Рис. 14. Діаграма розподілу атрибутів

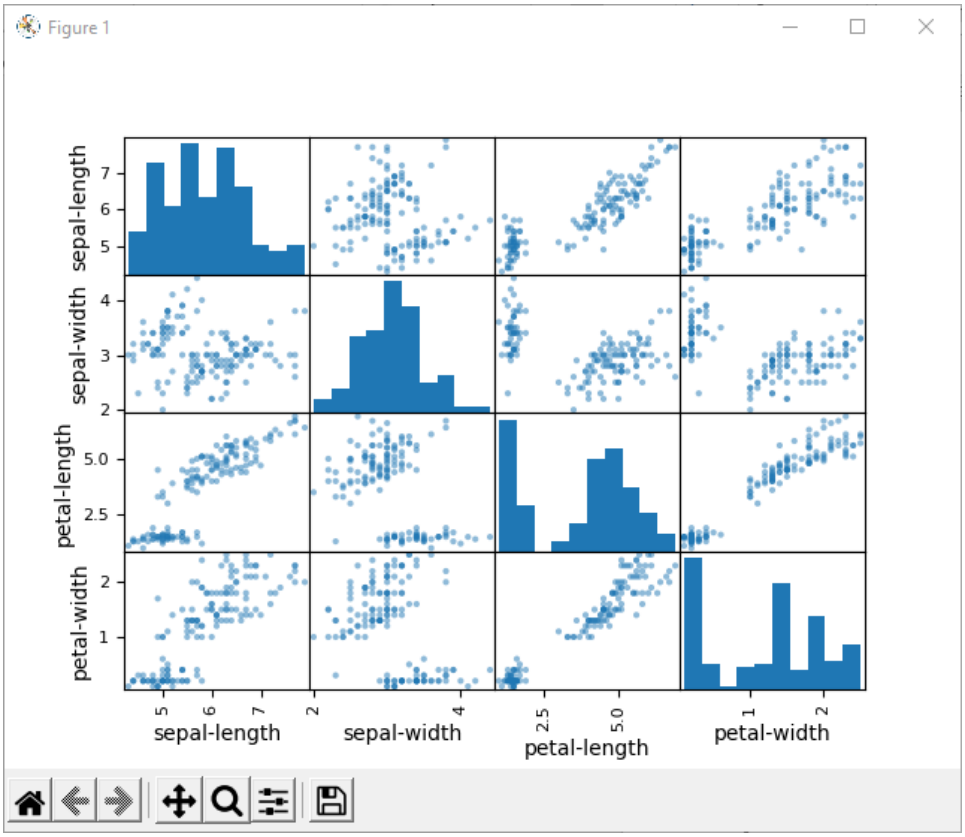

Рис. 15. Матриця діаграм розсіювання

```
# Завантаження датасету
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()
```

Рис. 16. Код програми

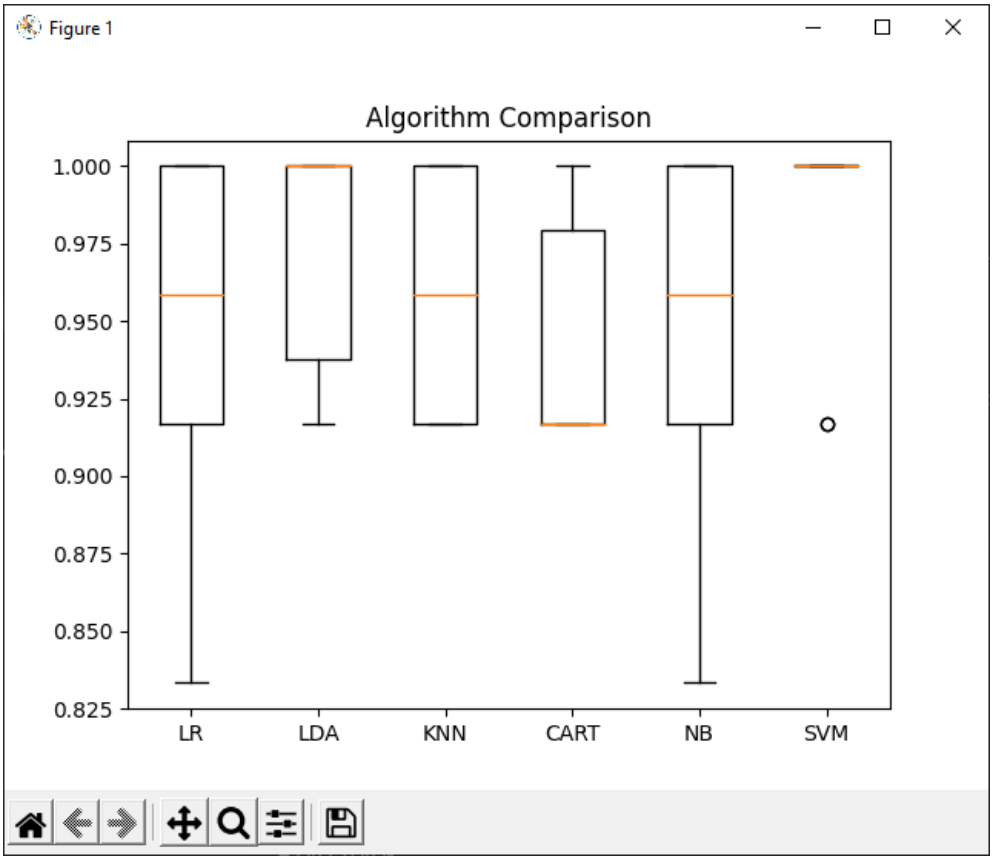

Рис. 17. Графік порівняння алгоритмів

```
# 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 = [('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
          ('LDA', LinearDiscriminantAnalysis()),
          ('KNN', KNeighborsClassifier()),
          ('CART', DecisionTreeClassifier()),
          ('NB', GaussianNB()),
          ('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='accuracy')
    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. 18. Код програми

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.038188)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

Рис. 19. Оцінка моделей

З даних вище можна зробити висновок про те, що найбільш оптимальним методом класифікації є SVM

	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. 20. Показники якості

```
# 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 массива X_new: {}".format(X_new.shape))
prediction = model.predict(X_new)
print("Прогноз: {}".format(prediction))
print("Спрогнозированная метка: {}".format(prediction[0]))
```

Рис. 21. Код програми

```
Прогноз: ['Iris-setosa']
Спрогнозированная метка: Iris-setosa
```

Рис. 22. Прогноз

Отже, за даними, отриманими в результаті виконання програми, можна побачити, що прогноз має точність 97%, а квітка з заданими параметрами належить до класу Iris-setosa

Завдання 2.4. Порівняння якості класифікаторів для набору даних завд. 2.1.

```
from methods import get_X_y
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.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

X, y, label_encoder = get_X_y()

# 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 = [('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
          ('LDA', LinearDiscriminantAnalysis()),
          ('KNN', KNeighborsClassifier()),
          ('CART', DecisionTreeClassifier()),
          ('NB', GaussianNB()),
          ('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='accuracy')
        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()

    # - SVC
    model = SVC(gamma='auto')
    model.fit(X_train, Y_train)
    predictions = model.predict(X_validation)
    print('SVC')
    print(classification_report(Y_validation, predictions))

    # - Gaussian
    model = GaussianNB()
    model.fit(X_train, Y_train)
    predictions = model.predict(X_validation)
    print('Gaussian')
    print(classification_report(Y_validation, predictions))

    # - DecisionTree
    model = DecisionTreeClassifier()
    model.fit(X_train, Y_train)
    predictions = model.predict(X_validation)
    print('DecisionTree')
    print(classification_report(Y_validation, predictions))
```

Рис. 23. Код програми

```
# - KNeighbors
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print('KNeighbors')
print(classification_report(Y_validation, predictions))

# - LinearDiscriminantAnalysis
model = LinearDiscriminantAnalysis()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print('LinearDiscriminantAnalysis')
print(classification_report(Y_validation, predictions))

# - LogisticRegression
model = LogisticRegression(solver='liblinear', multi_class='ovr')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print('LogisticRegression')
print(classification_report(Y_validation, predictions))
```

Рис. 24. Код програми

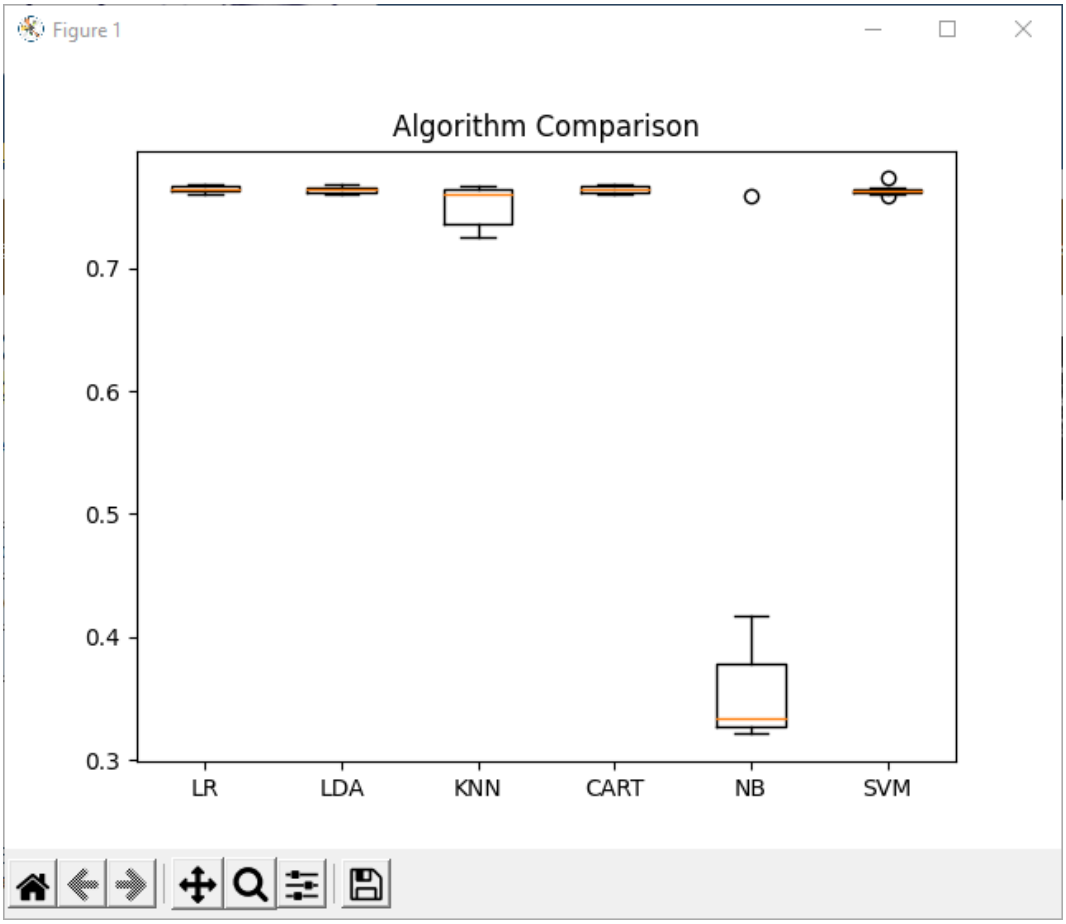

Рис. 25. Графік порівняння алгоритмів

```
LR: 0.764143 (0.002261)
LDA: 0.763604 (0.002613)
KNN: 0.751668 (0.016548)
CART: 0.763936 (0.002371)
NB: 0.388244 (0.127274)
SVM: 0.763397 (0.003756)
```

Рис. 26. Оцінка моделей

SVC

	precision	recall	f1-score	support
0	0.75	1.00	0.86	4483
1	0.87	0.05	0.10	1550
accuracy			0.76	6033
macro avg	0.81	0.53	0.48	6033
weighted avg	0.78	0.76	0.66	6033

Рис. 27. Показники якості для алгоритму SVC

Gaussian

	precision	recall	f1-score	support
0	0.91	0.13	0.22	4483
1	0.28	0.96	0.43	1550
accuracy			0.34	6033
macro avg	0.59	0.54	0.33	6033
weighted avg	0.75	0.34	0.28	6033

Рис. 28. Показники якості для алгоритму Gaussian

DecisionTree

	precision	recall	f1-score	support
0	0.76	0.98	0.86	4483
1	0.64	0.12	0.20	1550
accuracy			0.76	6033
macro avg	0.70	0.55	0.53	6033
weighted avg	0.73	0.76	0.69	6033

Рис. 29. Показники якості для алгоритму DecisionTree

KNeighbors

	precision	recall	f1-score	support
0	0.76	0.98	0.86	4483
1	0.64	0.12	0.20	1550
accuracy			0.76	6033
macro avg	0.70	0.55	0.53	6033
weighted avg	0.73	0.76	0.69	6033

Рис. 30. Показники якості для алгоритму KNeighbors

LinearDiscriminantAnalysis

	precision	recall	f1-score	support
0	0.76	0.97	0.86	4483
1	0.63	0.13	0.22	1550
accuracy			0.76	6033
macro avg	0.70	0.55	0.54	6033
weighted avg	0.73	0.76	0.69	6033

Рис. 31. Показники якості для алгоритму LinearDiscriminantAnalysis

LogisticRegression

	precision	recall	f1-score	support
0	0.76	0.98	0.86	4483
1	0.64	0.11	0.19	1550
accuracy			0.76	6033
macro avg	0.70	0.55	0.53	6033
weighted avg	0.73	0.76	0.69	6033

Рис. 32. Показники якості для алгоритму LogisticRegression

Беручи до розсуду графік, оцінки та результати показників якості можна виокремити три алгоритми: SVC, LinearDiscriminantAnalysis та LogisticRegression. Проте, судячи з графіку, можна сказати, що найкращим буде SVC.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge.

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

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)
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))

sns.set()
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")
```

Рис. 33. Код програми

```
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. 34. Результат виконання програми

Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають.

Перший параметр tol позначає точність рішення в цьому випадку це1e-2.

Другий параметр solver метод, який використовується у обчислювальних процесах, встановлений як ‘sag’ – ‘Stochastic Average Gradient’.

Опишіть які показники якості використовуються та їх отримані результати. Вставте у звіт та поясніть зображення Confusion.jpg

Були обраховані значення акуратності, точності, повноти та F-міра (рис. 34).

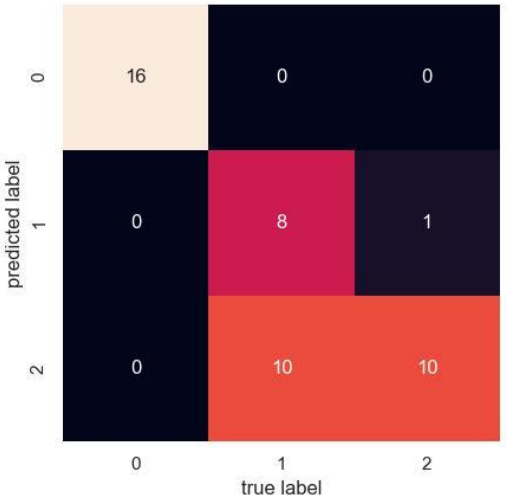

Рис. 35 – Confusion.jpg

На цьому зображенні показано скільки разів передбачений результат співпадав з реальним. Так, 16 разів перебачено було значення 0, яке є правдивим. Жодного разу при передбаченому 0 не було виявленого хибного результату. При передбаченому результаті 1 виявлено одну помилку. При передбаченому результаті 2 половина результатів є правдивими, а половина – хибними.

Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.

Коефіцієнт Коена Каппа – статистика, яка показує коефіцієнт зменшення помилок між класифікацією та випадковою класифікацією. У цьому випадку маємо значення 0,6431 > 0,5, що означає більш-менш задовільну згоду між коефіцієнтами.

Коефіцієнт кореляції Метьюза – міра якості моно- та мультикласових класифікацій. Отримане значення 0,6831, що означає, що класифікатор близький до правильного результату.

Висновок: в ході виконання лабораторної роботи було досліджено різні методи класифікації даних та навчено їх порівнювати за допомогою мови програмування Python.

GitHub