**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 №1:**

|  |  |
| --- | --- |
| Ознака | Можливі значення |
| age | continuous |
| workclass | Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked |
| fnlwgt | continuous |
| education | Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool |
| education-num | continuous |
| marital-status | Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse |
| occupation | Tech-support, Craft-repair, Other-service, Sales, Execmanagerial, Prof-specialty, Handlers-cleaners, Machineop-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces |
| relationship | Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried |
| race | White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black |
| sex | Female, Male |
| capital-gain | continuous |
| capital-loss | continuous |
| hours-per-week | continuous |
| native-country | United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-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, Trinadad&Tobago, Peru, Hong, Holand-Netherlands |

**Напишемо код для класифікації:**

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

from sklearn.model\_selection import cross\_val\_score

# Input file containing data

input\_file = 'income\_data.txt'

# Read the data

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

# Convert to numpy array

X = np.array(X)

# Convert string data to numerical data

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)

# Create SVM classifier

classifier = OneVsOneClassifier(LinearSVC(random\_state=0, dual=False))

# Train the classifier

classifier.fit(X, y)

# Cross validation

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, dual=False))

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

# Compute the F1 score of the SVM classifier

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

# Predict output for a test datapoint

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',

              'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'UnitedStates']

# Encode test datapoint

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).reshape(1, -1)

# Run classifier on encoded datapoint and print output

predicted\_class = classifier.predict(input\_data\_encoded)

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат:

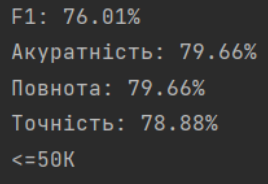


Рис. 1. Результати класифікатора щорічного прибутку

Задана точка відноситься до класифікатора «<=50K»

Завдання №2:

Код:

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.svm import SVC

from sklearn.multiclass import OneVsOneClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

# Input file containing data

input\_file = 'income\_data.txt'

# Read the data

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

# Convert to numpy array

X = np.array(X)

# Convert string data to numerical data

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)

# Create SVM classifier

classifier = OneVsOneClassifier(SVC(kernel='sigmoid')) // kernel = 'poly' kernel = 'rbf'

# Train the classifier

classifier.fit(X, y)

# Cross validation

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=5)

classifier = OneVsOneClassifier(SVC(kernel='sigmoid')) // kernel = 'poly' kernel = 'rbf'

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

# Compute the F1 score of the SVM classifier

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

# Predict output for a test datapoint

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',

              'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'UnitedStates']

# Encode test datapoint

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).reshape(1, -1)

# Run classifier on encoded datapoint and print output

predicted\_class = classifier.predict(input\_data\_encoded)

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат:

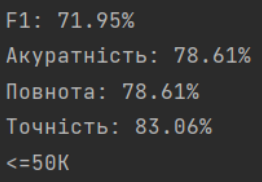


Рис. 2. Результати SVM з гаусовим ядром

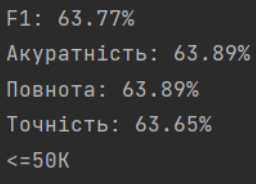


Рис. 3. Результати SVM з сигмоїдальним ядром

Завдання №3:

Ознайомлення зі структурою даних та результати:

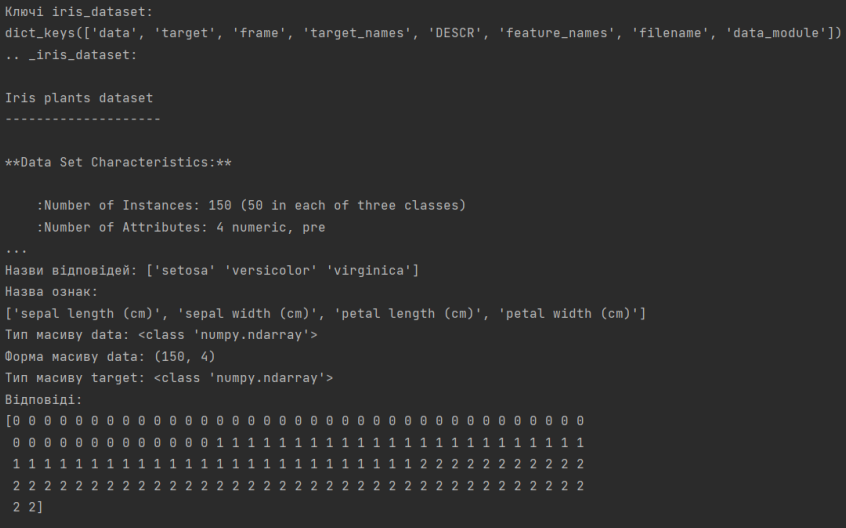


Рис. 4. Ознайомлення зі структурою даних та результати

Код візуалізації:

from pandas import read\_csv

from pandas.plotting import scatter\_matrix

from matplotlib import pyplot

# Load dataset

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)

# box and whisker plots

dataset.plot(kind='box', subplots=True, layout=(

    2, 2), sharex=False, sharey=False)

pyplot.show()

# histograms

dataset.hist()

pyplot.show()

# scatter plot matrix

scatter\_matrix(dataset)

pyplot.show()

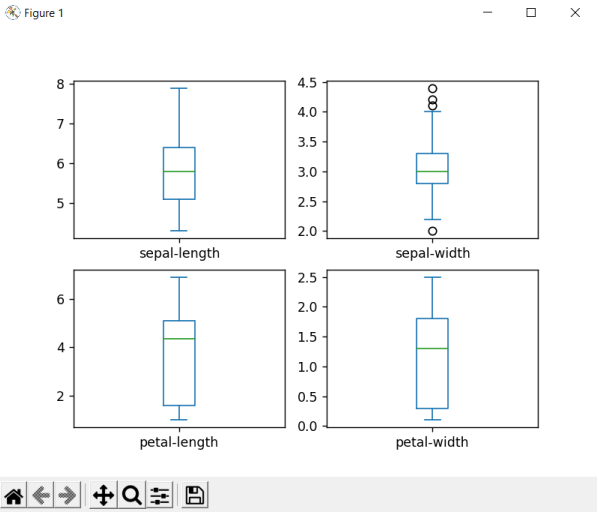


Рис. 5. Одновимірний графік

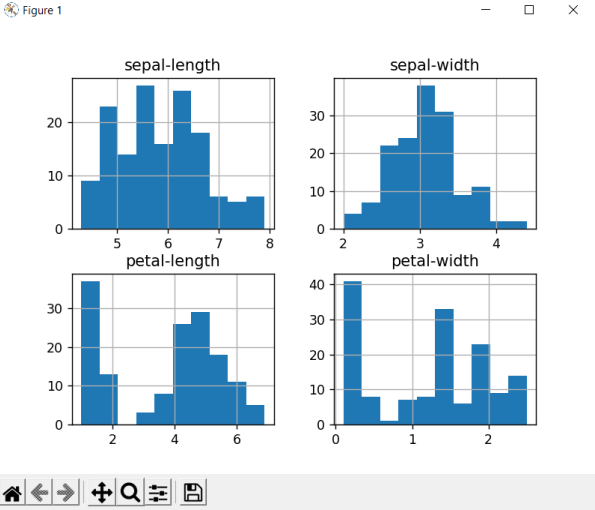


Рис. 6. Гістограма розподілу атрибутів датасета

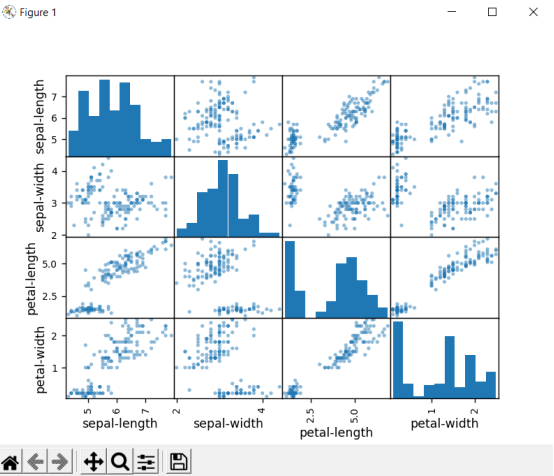


Рис. 7. Багатовимірний графік

Протестуємо 6 різних алгоритмів:

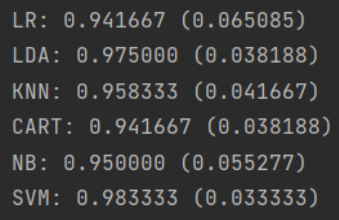


Рис. 8. Багатовимірний графік

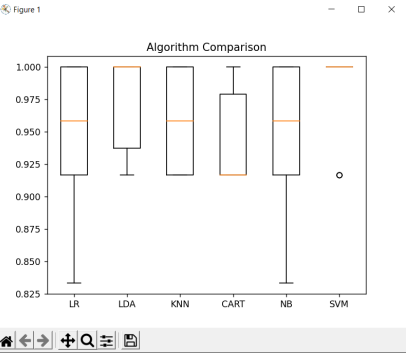


Рис. 9. Багатовимірний графік

З рисунків бачимо, що найбільш точним алгоритмом у цьому випадку є SVM. Написаний код для здійснення прогнозу:

X\_new = np.array([[5, 2.9, 1, 0.2]])

print("X\_new.shape: {}".format(X\_new.shape))

prediction = model.predict(X\_new)

print("Prediction of Species: {}".format(prediction))

Отримуємо:

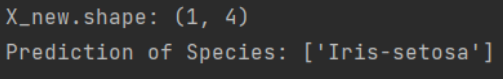


Рис. 10. Отриманий клас

Отже, спрогнозований сорт ірису – Iris-setosa.

Завдання №4:

Порівняймо алгоритми з минулого завдання, використовуючи задачу з першого завдання. Код:

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.svm import SVC

from sklearn.multiclass import OneVsOneClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

# Input file containing data

input\_file = 'income\_data.txt'

# Read the data

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

# Convert to numpy array

X = np.array(X)

# Convert string data to numerical data

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)

# Cross validation

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

classifier = OneVsOneClassifier(LinearRegression())

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

print("")

print("LR:")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(LinearDiscriminantAnalysis())

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

print("")

print("LDA:")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(KNeighborsClassifier())

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

print("")

print("KNN:")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(DecisionTreeClassifier())

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

print("")

print("CART:")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(SVC())

classifier.fit(X\_train, y\_train)

y\_test\_pred = classifier.predict(X\_test)

print("")

print("SVM:")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1: " + str(round(100\*f1.mean(), 2)) + "%")

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Акуратність: " + str(round(100\*accuracy.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)

print("Повнота: " + str(round(100\*recall.mean(), 2)) + "%")

precision = cross\_val\_score(

    classifier, X, y, scoring='precision\_weighted', cv=3)

print("Точність: " + str(round(100\*precision.mean(), 2)) + "%")

X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5], [6, 5], [5.6, 5], [3.3,

                                                                             0.4], [3.9, 0.9], [2.8, 1], [0.5, 3.4], [1, 4], [0.6, 4.9]])

y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])

# Create the logistic regression classifier

classifier = linear\_model.LogisticRegression(solver='liblinear', C=1)

# classifier = linear\_model.LogisticRegression(solver='liblinear', C=100)

# Train the classifier

classifier.fit(X, y)

# Visualize the performance of the classifier

visualize\_classifier(classifier, X, y)

Результат:

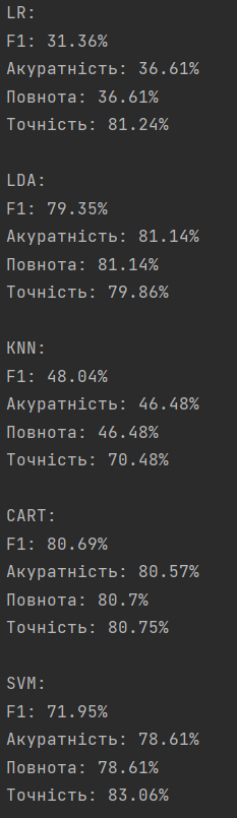


Рис. 11. Результати порівняння алгоритмів

Отже, переглянувши всі результати, робимо висновок, що для цієї задачі найбільше підійде алгоритм CART.

Завдання №5:

Виконаймо класифікацію даних лінійним класифікатором Ridge. Код:

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

from sklearn import metrics

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.linear\_model import RidgeClassifier

from sklearn.metrics import confusion\_matrix

from io import BytesIO  # neded for plot

import seaborn as sns

sns.set()

iris = load\_iris()

X, y = iris.data, iris.target

Xtrain, X\_test, 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(X\_test)

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")

# Save SVG in a fake file object.

f = BytesIO()

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

Результат:

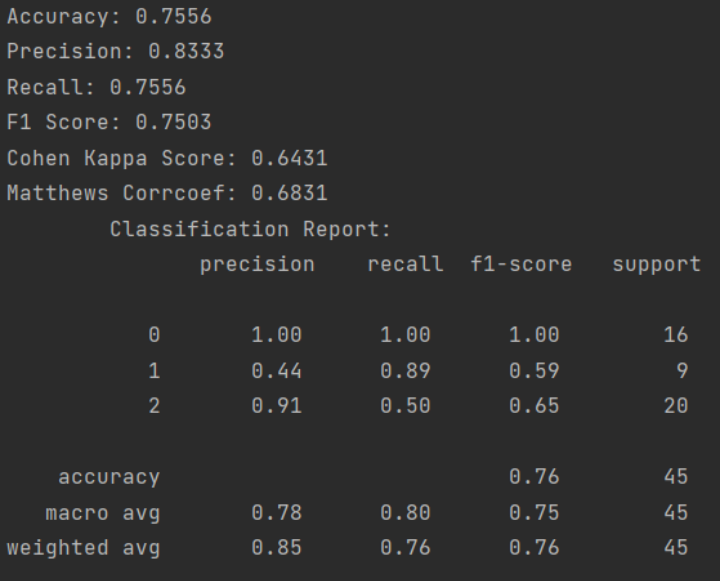


Рис. 12. Класифікація даних лінійним класифікатором Ridge

У класифікаторі Ridge були застосовані налаштування tol та solver. Параметр tol задається для контролю точності рішення, тоді як solver використовується в алгоритмах обчислення. Для оцінювання якості класифікатора використовуються наступні критерії:

* Акуратність: відображає загальну точність класифікації.
* Точність: показує, наскільки точно модель ідентифікує позитивні випадки.
* Повнота: вимірює здатність моделі виявляти всі позитивні випадки.
* F1-показник: комбінує точність і повноту в один показник, що дає збалансовану оцінку якості.
* Коефіцієнт каппи Коена: статистичний показник, який вимірює надійність між двома оцінювачами для категорійних даних, з можливими значеннями від 0 до 1.
* Коефіцієнт кореляції Метьюза: використовується для визначення сили зв'язку між двома бінарними змінними, з можливими значеннями від 0 до

Ці метрики в сукупності дозволяють всебічно оцінити ефективність і надійність класифікатора Ridge.

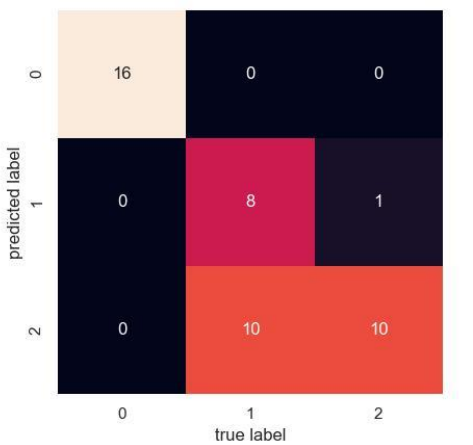


Рис. 13. Матриця з показниками якості

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