**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.

Завдання 2.1

Опис набору вхідних даних:

1. Age (вік): 17 – 90 років

2. Workclass (зайнятість): Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked

3. Fnlwgt (Final weight, ваговий коефіцієнт): числове значення

4. Education (освіта): Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc- acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool

5. Education-num (числове преставлення рівня освіти): числове значення

6. Marital-status (сімейний стан): Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse

7. Occupation (професія): Tech-support, Craft-repair, Other-service, Sales, Exec- managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed- Forces

8. Relationship (відносити): Wife, Own-child, Husband, Not-in-family, Other- relative, Unmarried

9. Race (раса): White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

10. Sex (стать): Female, Male

11. Capital-gain (капіталовигода): числове значення

12. Capital-loss (капіталовтрата): числове значення

13. Hours-per-week (години на тиждень): числове значення

14. Native-country (Батьківщина): 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.

15. Клас доходу: >=50К чи <= 50К.

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, 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)  
  
scaller = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
X = scaller.fit\_transform(X)  
  
# Create SVM classifier  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
  
# Train the classifier  
classifier.fit(X=X, y=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))  
  
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)  
  
# Compute the F1 score of the SVM classifier  
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)) + "%")  
f1\_values = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")  
  
print("F1 score: " + str(round(100\*f1.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', 'United-States']  
  
# Encode test datapoint  
input\_data\_encoded = np.array([-1] \* len(input\_data))  
count = 0  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = item  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([item]))  
 count += 1  
input\_data\_encoded = input\_data\_encoded.astype(int)  
input\_data\_encoded = [input\_data\_encoded]  
  
  
# Run classifier on encoded datapoint and print output  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

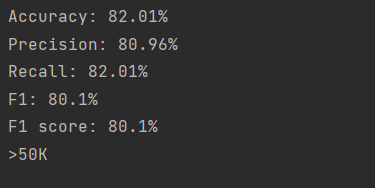


Рис.1 Результат виконання

Завдання 2.2

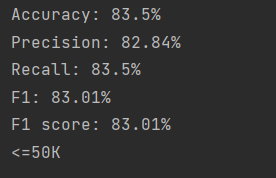


Рис. 2 Результат виконання (poly ядро)

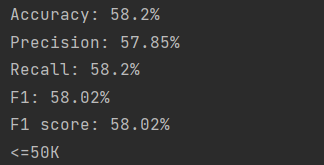
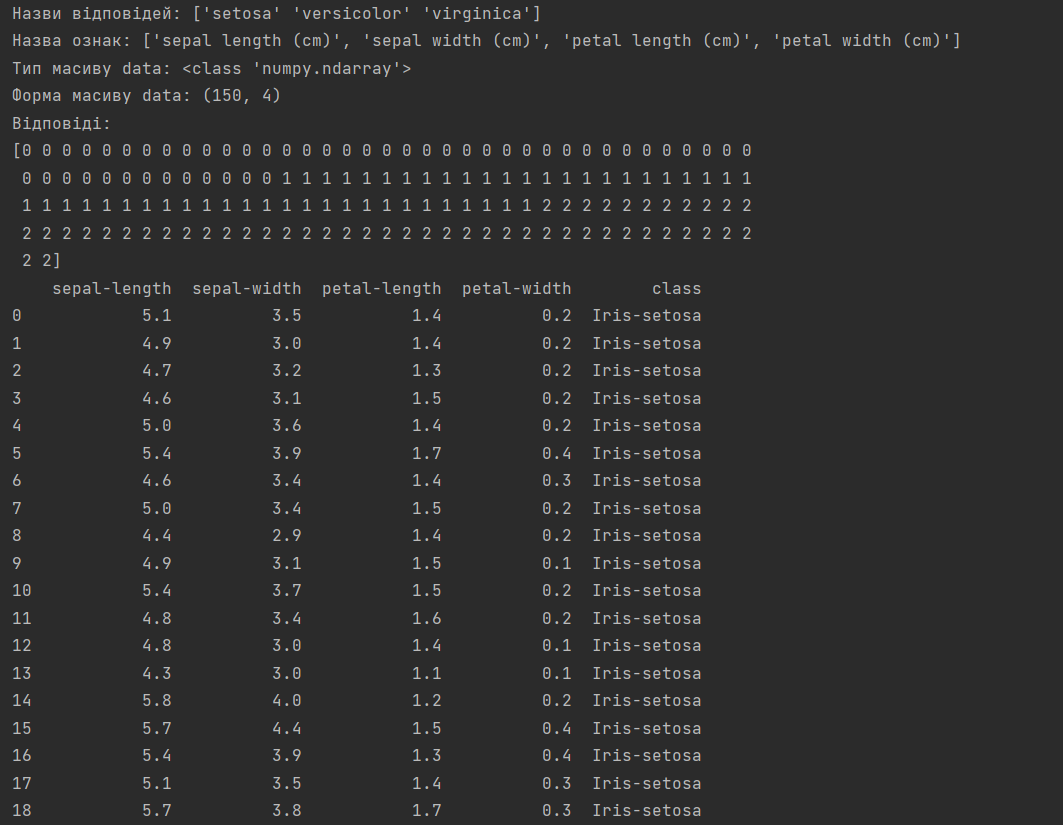


Рис.3 Результат виконання (sigmoid ядро)

У результаті визначено, що ядро RBF продемонструвало добрі результати, маючи невеликий недолік порівняно з поліноміальним ядром, але виграючи у швидкості. З іншого боку, сигмоїдне ядро показало менший результат. З урахуванням цього висновку можна стверджувати, що для нашого конкретного випадку оптимальним вибором буде використання RBF-ядра, яке комбінує високу точність і ефективність.

Завдання 2.3

from sklearn.datasets import load\_iris  
import numpy as np  
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  
  
iris\_dataset = load\_iris()  
print(f'Ключі iris\_dataset: {iris\_dataset.keys()}')  
print(iris\_dataset['DESCR'][:193] + "\n....")  
print(f"Назви відповідей: {iris\_dataset['target\_names']}")  
print(f"Назва ознак: {iris\_dataset['feature\_names']}")  
print(f"Тип масиву data: {type(iris\_dataset['data'])}")  
print(f"Форма масиву data: {iris\_dataset['data'].shape}")  
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))  
  
# 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)  
  
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')))  
  
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()  
  
# 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]])  
  
for name, model in models:  
 model.fit(X\_train, Y\_train)  
 prediction = model.predict(X\_new)  
 print("Прогноз: {}".format(prediction))  
 print(accuracy\_score(Y\_validation, predictions))  
 print(confusion\_matrix(Y\_validation, predictions))  
 print(classification\_report(Y\_validation, predictions))

  
Рис.4 Результат виконання

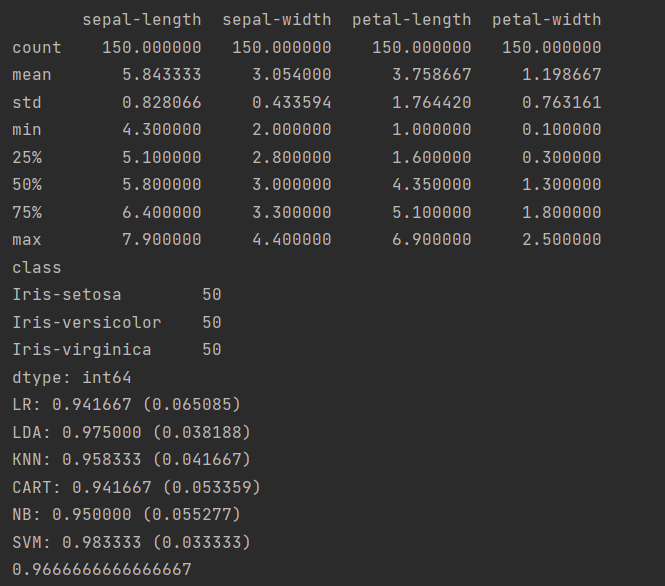


Рис.5 Результат виконання

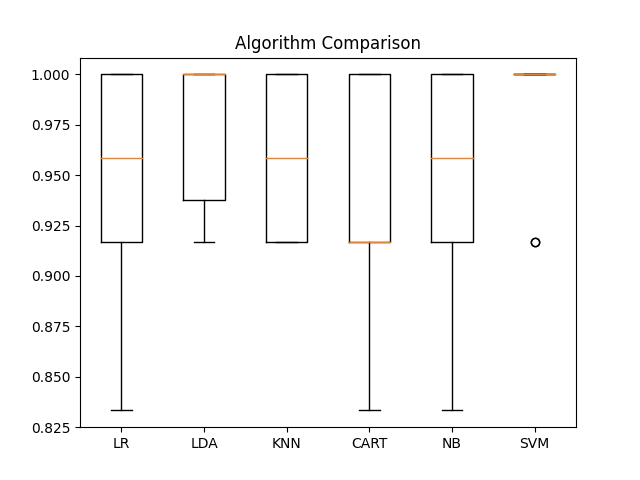


Рис. 6 Рисунок порівняння алгоритмів

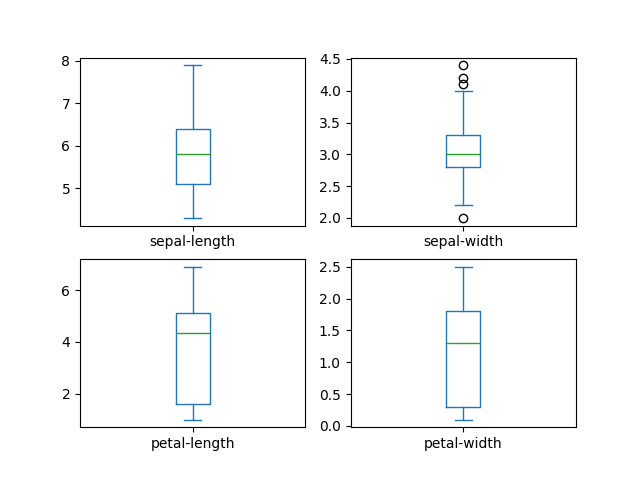


Рис. 7 Результат діаграми розмаху

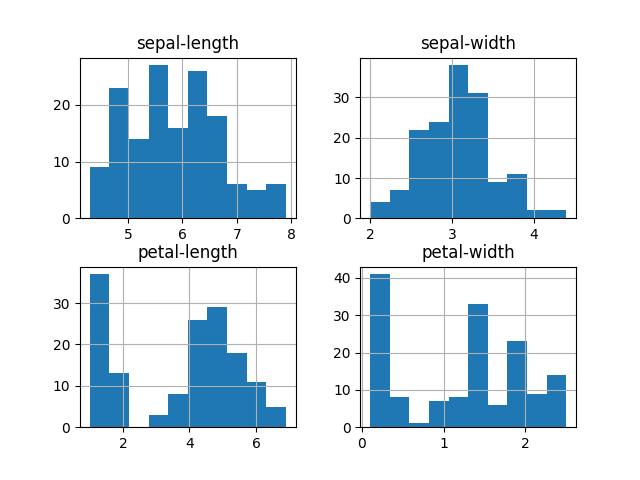


Рис. 8 Гістрограма розподілу атрибутів

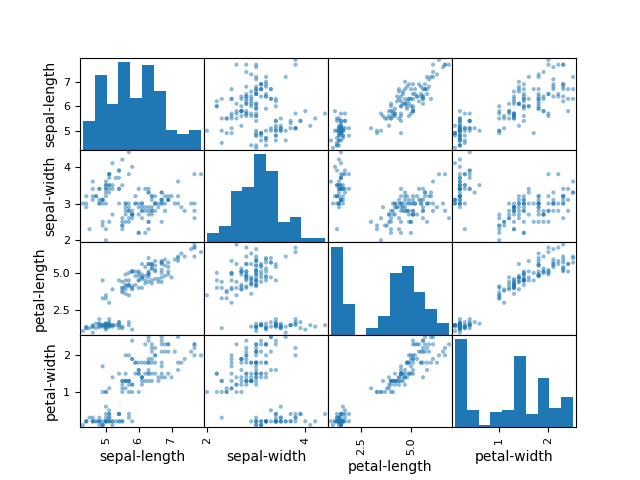


Рис. 9 Матриця діаграми розсіювання

Квітка належала до класу Iris-setosa

Узагальнюючи з діаграм, можемо визначити, що найбільш ефективною виявилася модель лінійного дискримінантного аналізу, хоча вона має певні недоліки, такі як нестійкість та більший час виконання під час тестування.

Завдання 2.4

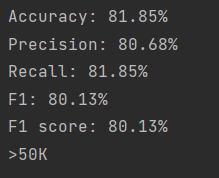


Рис.10 Точність класифікатора LR

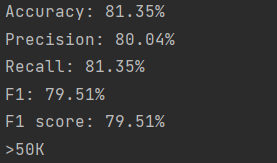


Рис. 11 Точність класифікатора LDA

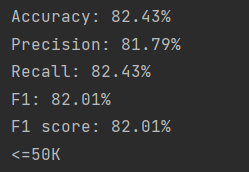


Рис. 12 Точність класифікатора KNN

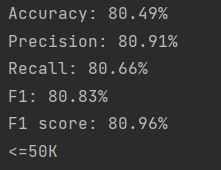


Рис. 13 Точність класифікатора CART

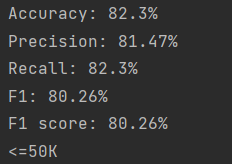


Рис. 14 Точність класифікатора SVM

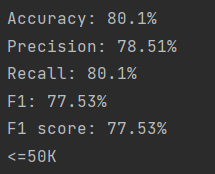


Рис. 15 Точність класифікатора NB

Завдання 2.5

import numpy as np  
import seaborn as sns  
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  
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")  
# Save SVG in a fake file object.  
f = BytesIO()  
plt.savefig(f, format="svg")

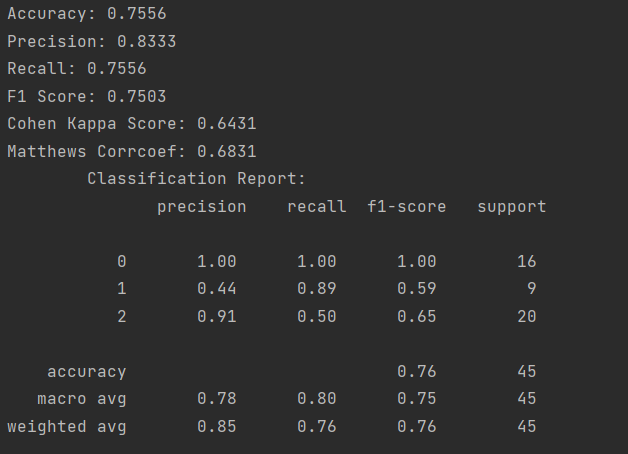


Рис.16 Результат виконання

У використаному класифікаторі Ridge встановлені наступні налаштування: tol = 1e-2 та solver = "sag". Параметр tol визначає критерій збіжності оптимізації, де оптимізація припиниться, коли різниця між послідовними ітераціями стане меншою за 1e-2. Збільшення tol може прискорити збіжність, але може вплинути на точність результату. solver = "sag" вказує метод вирішення задачі оптимізації, в даному випадку — Stochastic Average Gradient Descent, який застосовується до логістичної регресії.

та 4. Для оцінки моделі використовуються показники якості, такі як accuracy, precision, recall, f1 score, Cohen Kappa score, Metthews Corrcoef.

a. Accuracy визначає долю правильних класифікацій і становить 0.7556.

b. Precision — співвідношення правильних позитивних класифікацій до всіх класифікацій — 0.8333.

c. Recall — співвідношення правильних позитивних класифікацій до загального числа істинних позитивних екземплярів — 0.7556.

d. F1 score — гармонійне середнє точності та повноти — 0.7503.

e. Cohen Kappa score — статистична міра збігу між класифікатором та реальними даними, з урахуванням випадкового збігу — 0.6431.

f. Matthews Corrcoef — міра якості бінарної класифікації, яка враховує і істинно позитивні, і істинно негативні випадки — значення від -1 до 1, де 1 — ідеальна згода, 0 — випадковий збіг, -1 — повна незгода.

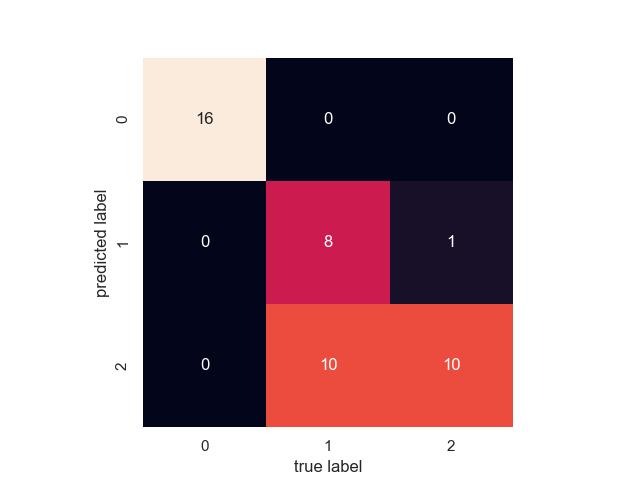


Рис.17 Результат виконання

3. Зображення є карткою матриці помилок, яка відображає кількість правильно класифікованих екземплярів і кількість помилок для кожного класу. За додаванням значень по діагоналі можна визначити загальну кількість правильних класифікацій.

GitHub: https://github.com/unravee1/AI\_labs