**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:**  Класифікація за допомогою машин опорних векторів (SVM).

**Результат:**

Дані з https://archive.ics.uci.edu/ml/datasets/census+income:

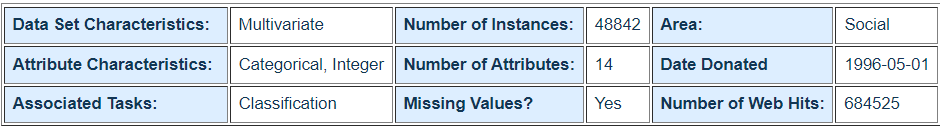


Рис. 1.1.

14 ознак набору даних:

- age: категоріальна  
 - workclass: категоріальна  
 - fnlwgt: числова  
 - education: категоріальна  
 - education-num: числова  
 - marital-status: категоріальна  
 - occupation: категоріальна  
 - relationship: категоріальна  
 - race: категоріальна  
 - sex: категоріальна  
 - capital-gain: числова  
 - capital-loss: числова  
 - hours-per-week: числова  
 - native-country: категоріальна

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:

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

input\_file = 'income\_data.txt'

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

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)

classifier = OneVsOneClassifier(LinearSVC(random\_state=0))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_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)) + "%")

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']

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)

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

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

**Завдання 2:**  Порівняння якості класифікаторів SVM з нелінійними ядрами.

**Результат:**

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 3:** Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

**Результат:**

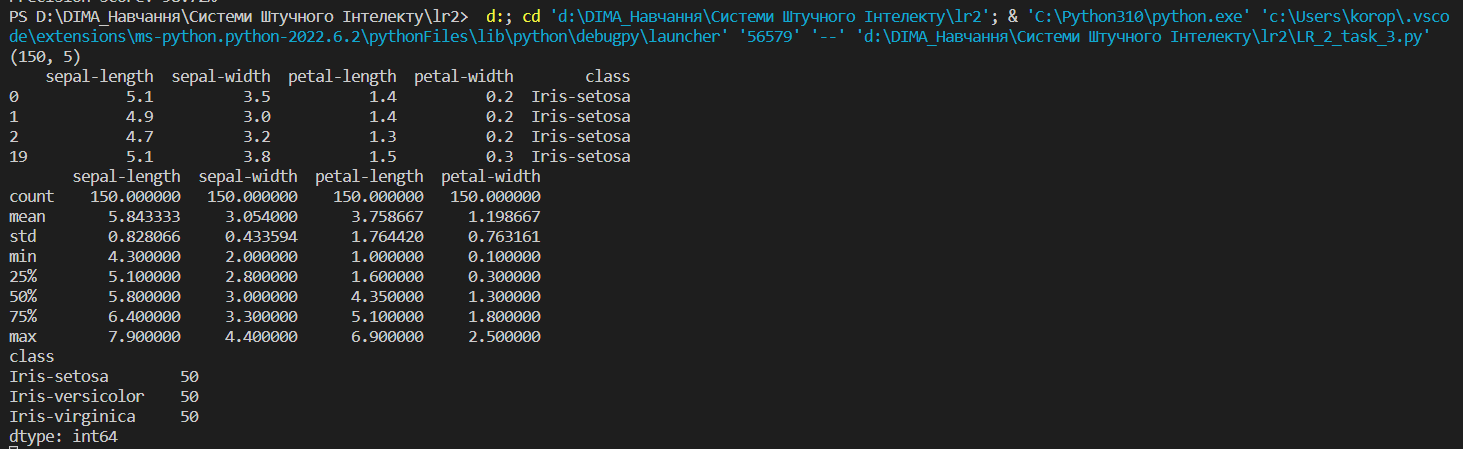


Рис. 3.1. Структура даних

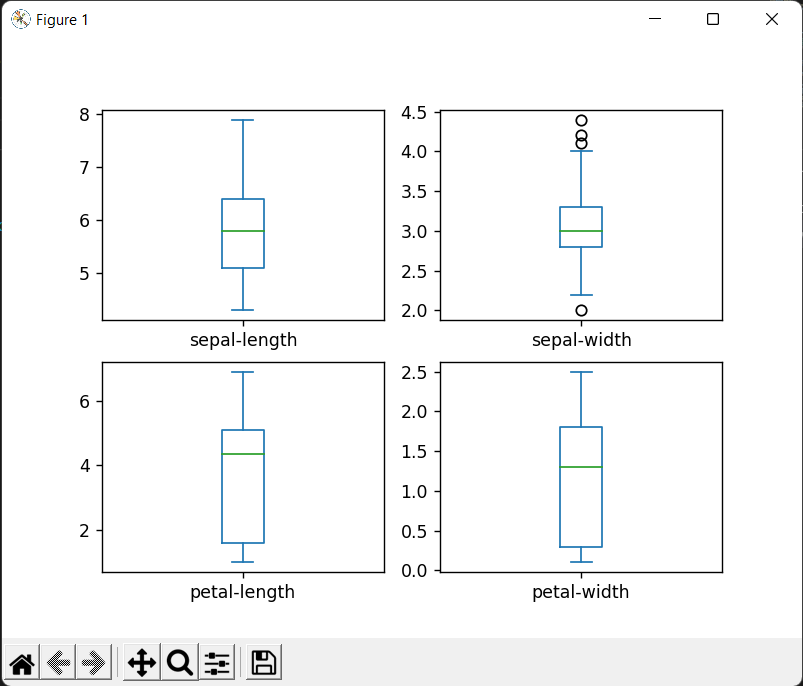


Рис. 3.2. Одновимірні графіки характеристик

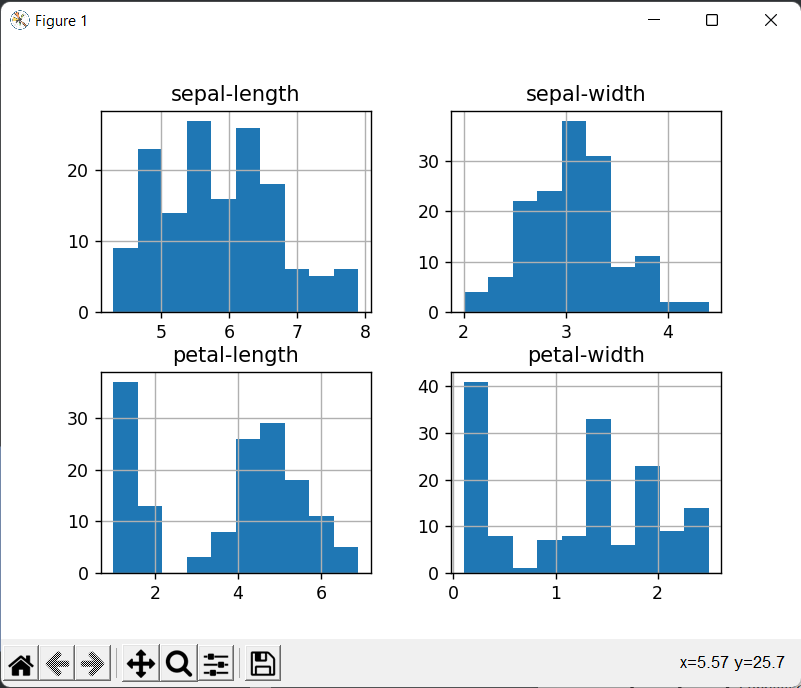


Рис. 3.3. Діаграма розмаху атрибутів

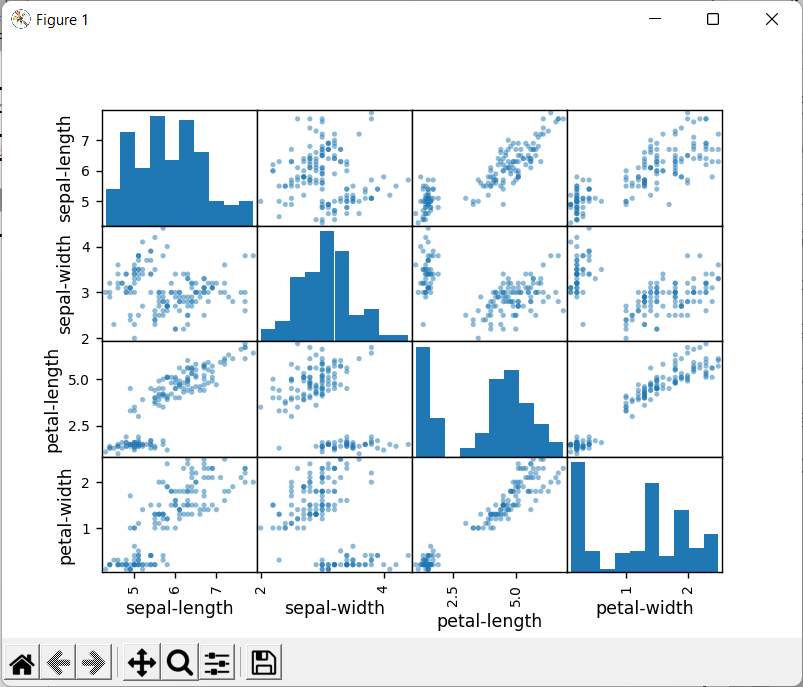


Рис. 3.4. Матриця розсіювання

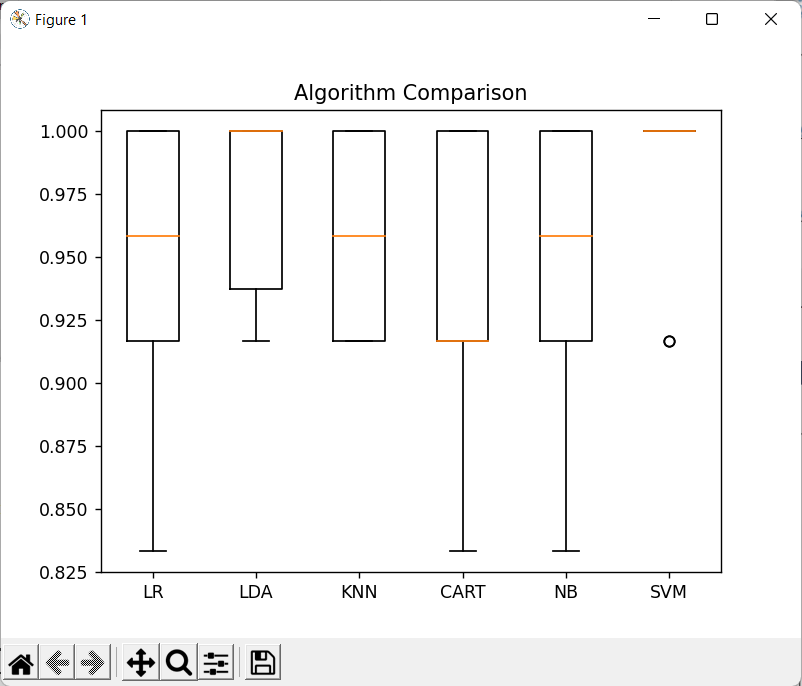


Рис. 3.5. Графік порівняння алгоритмів

Проаналізувавши ортимані графіки, я дійшов висновку, що кращим був обрав метод класифікації CVM, тому що він показав найвищу якість.

Код:

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)

print(dataset.shape)

print(dataset.head(20))

print(dataset.describe())

print(dataset.groupby('class').size())

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)

pyplot.show()

dataset.hist()

pyplot.show()

scatter\_matrix(dataset)

pyplot.show()

array = dataset.values

X = array[:,0:4]

y = array[:,4]

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()))

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(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]))

**Завдання 4:**  Порівняння якості класифікаторів для набору даних завдання 2.1.

**Результат:**

Код:

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

input\_file = 'income\_data.txt'

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

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)

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

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()))

pyplot.boxplot(results, labels=names)

pyplot.title('Algorithm Comparison')

pyplot.show()

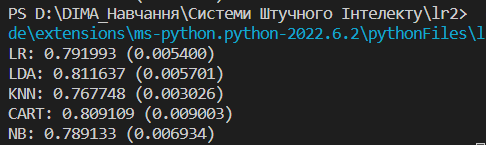


Рис. 4.1. Результат виконання

**Завдання 5:**  Класифікація даних лінійним класифікатором Ridge.

**Результат:**

Код:

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, 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))

mat = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(mat.T, square = True, annot = True, fmt = 'd', cbar = False)

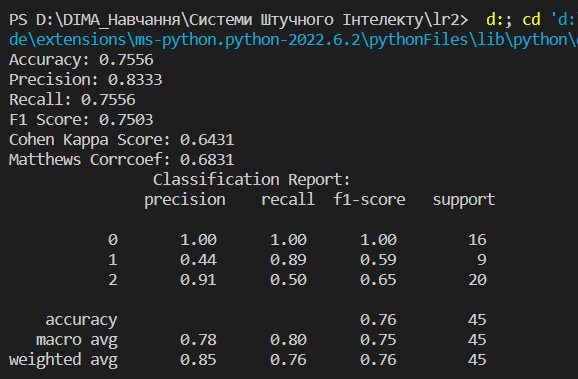
plt.xlabel('true label')

plt.ylabel('predicted label');

plt.savefig("Confusion.jpg")

f = BytesIO()

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



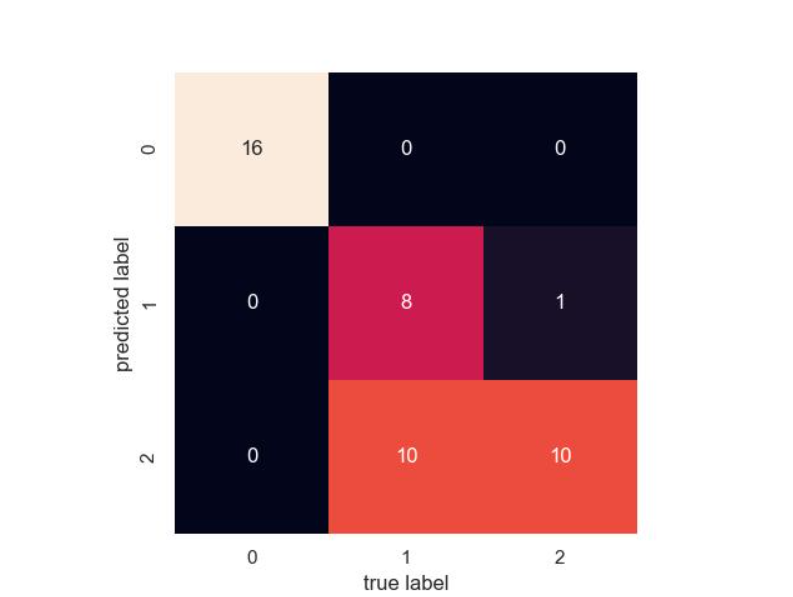


Рис. 5.1. Результат виконання

Класифікатор має наступні параметри:

- tol – точність класифікації

- solver – алгоритм, який виконує класифікацію

На зображені Confusion.jpg наведені результати класифікації.

На вертикалькій 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.

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

**Висновок:** на цій лабораторній роботі ми використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчилися їх порівнювати.