**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**

**Посилання на GitHub:** <https://github.com/Max2002/AI_IPZ-19-3_LMV>

**Завдання 1.** Класифікація за допомогою машин опорних векторів (SVM) Лістинг програми:

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  
from sklearn.model\_selection import 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(', ')  
 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  
  
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:  
 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 = OneVsOneClassifier(LinearSVC(random\_state=0))  
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)) + "%")  
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)) + "%")  
  
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 = 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]  
predicate\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicate\_class)[0])

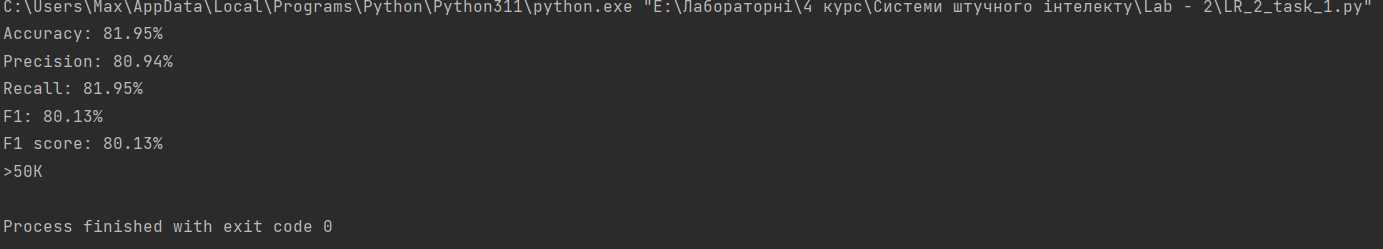


Рис. 1 Результат виконання

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

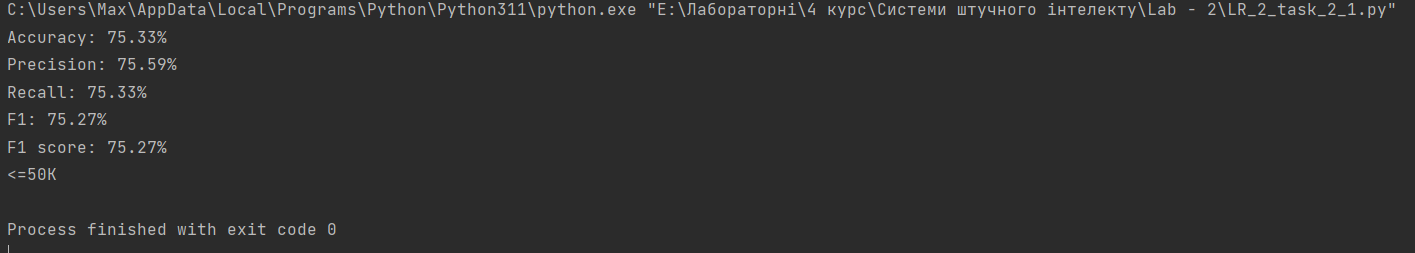


Рис. 2 Поліноміальне ядро

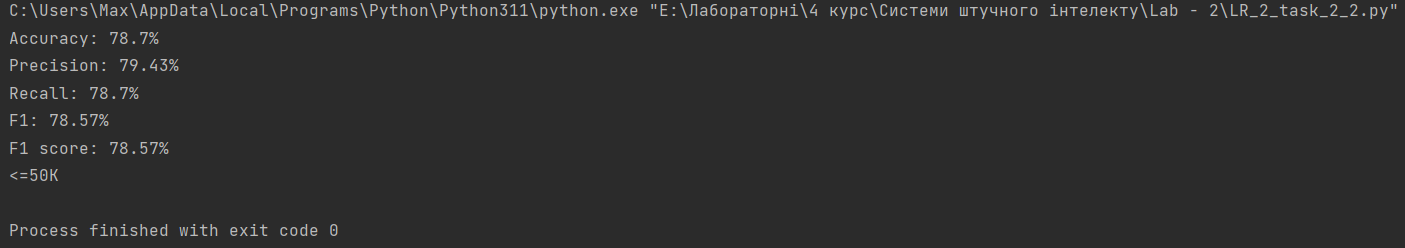


Рис. 3 Гаусове ядро

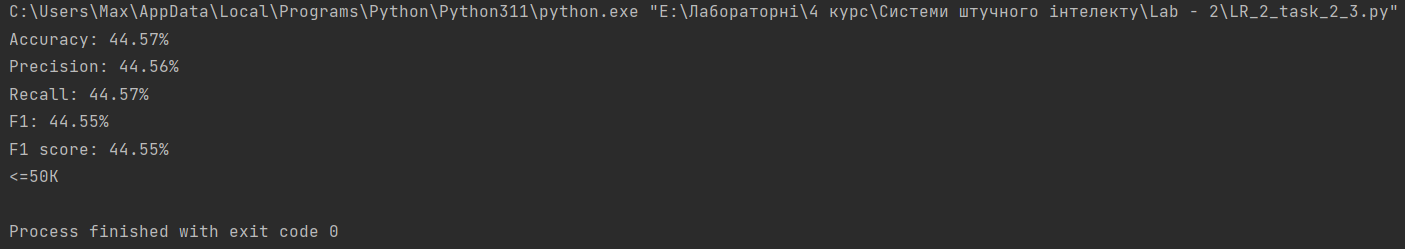


Рис. 4 Сигмоїдальне ядро

З результатів ми бачимо, що RFB має хороші результати, але гірший перед поліноміальним ядром, хоча і має переваги в швидкодії. Сигмоїдне ядро має більш низький результат. У висновоку можна зазначити, що для даного випадку краще підходить RBF, тому що має поєднання точності та швидкості.

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

Лістинг програми:

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("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))  
# 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))

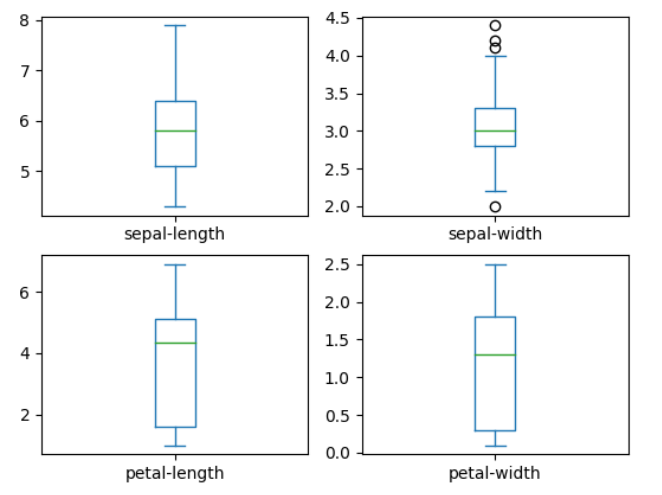


Рис. 5 Діаграма розмаху

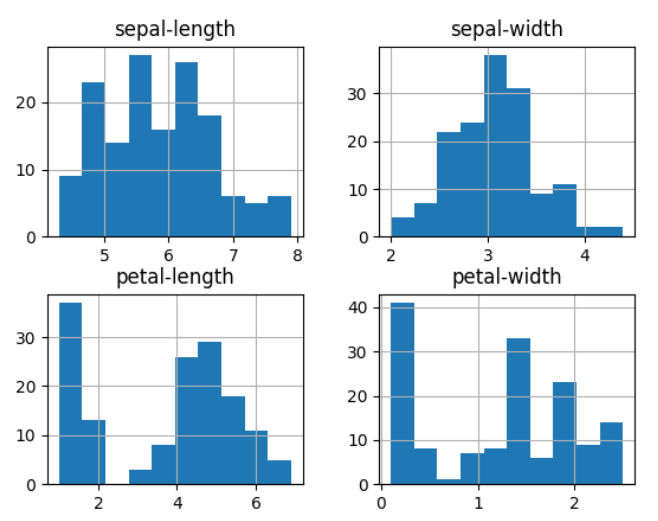


Рис. 6 Гістрограма розподілу атрибутів

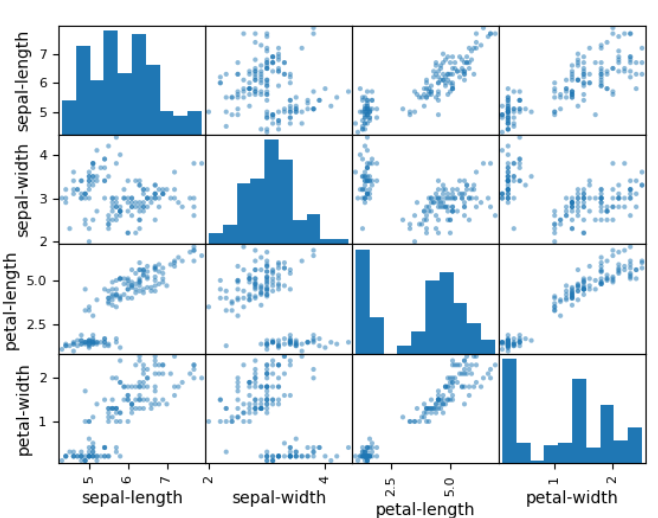


Рис. 7 Матриця діаграми розсіювання

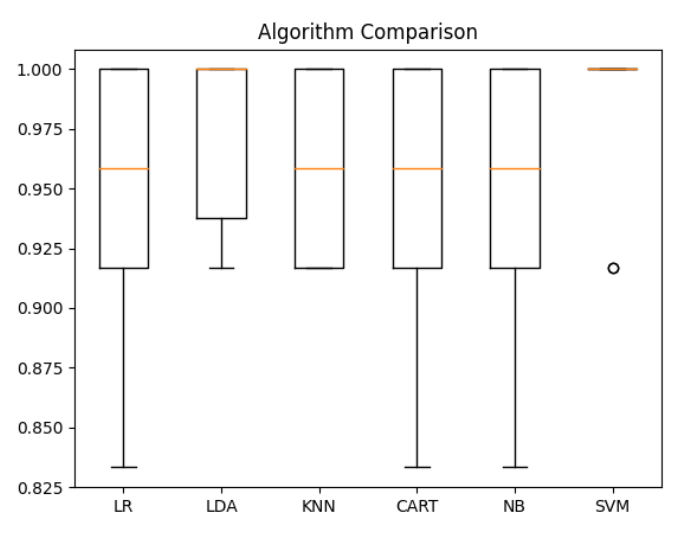


Рис. 8 Рисунок порівняння алгоритмів

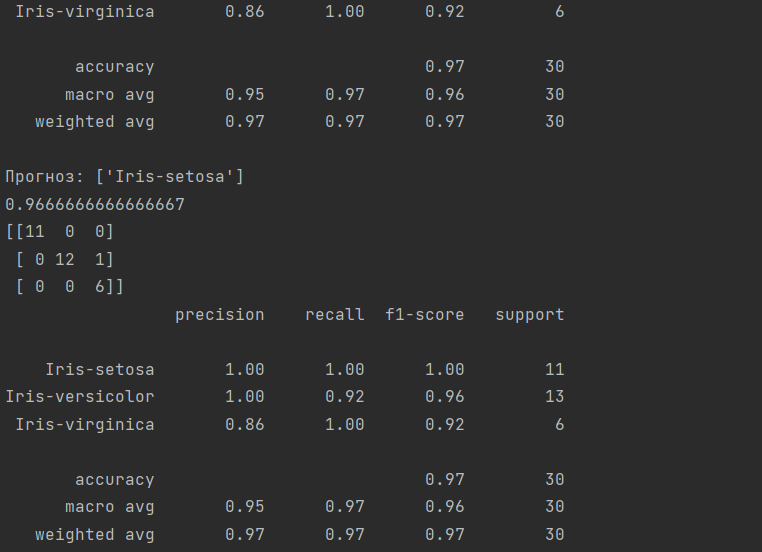
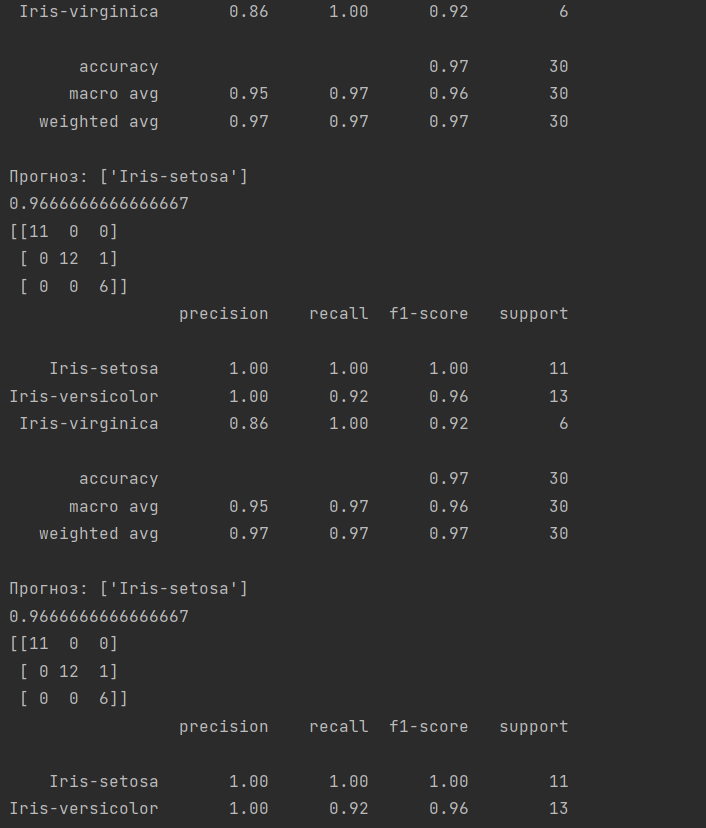
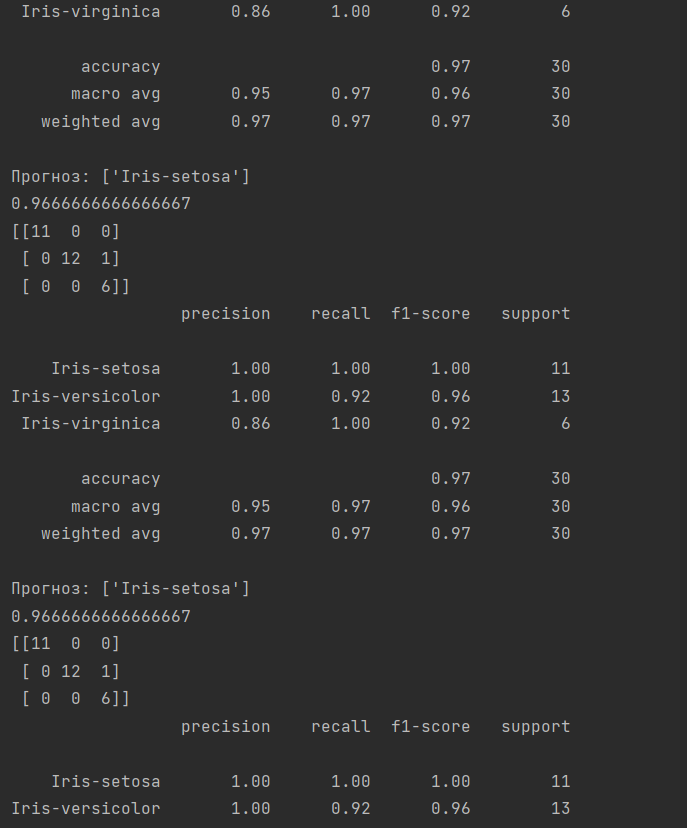
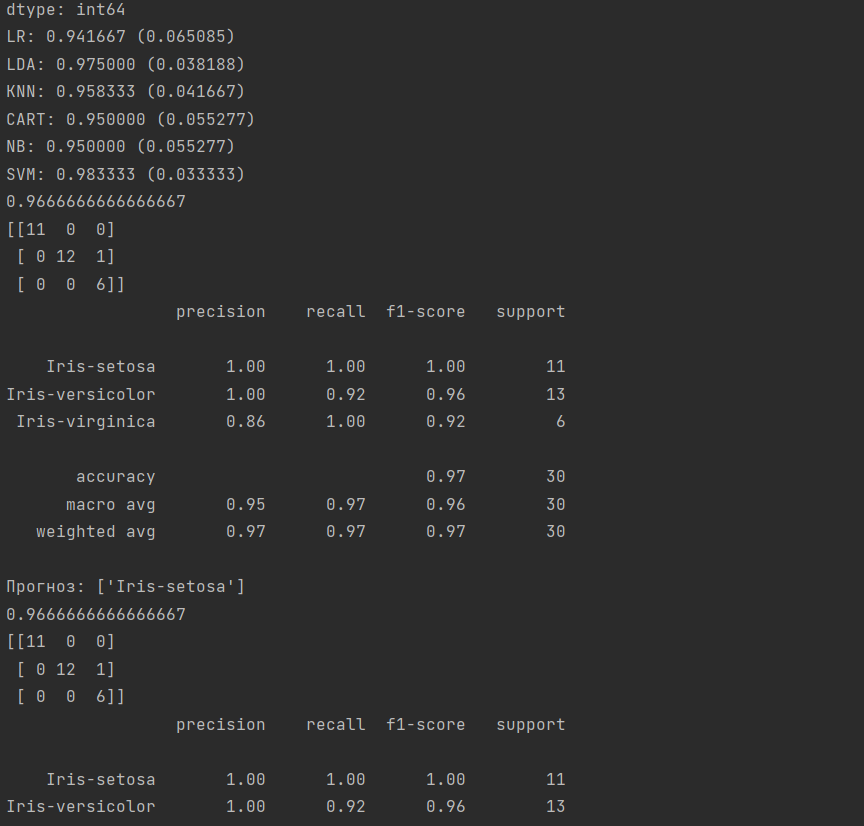
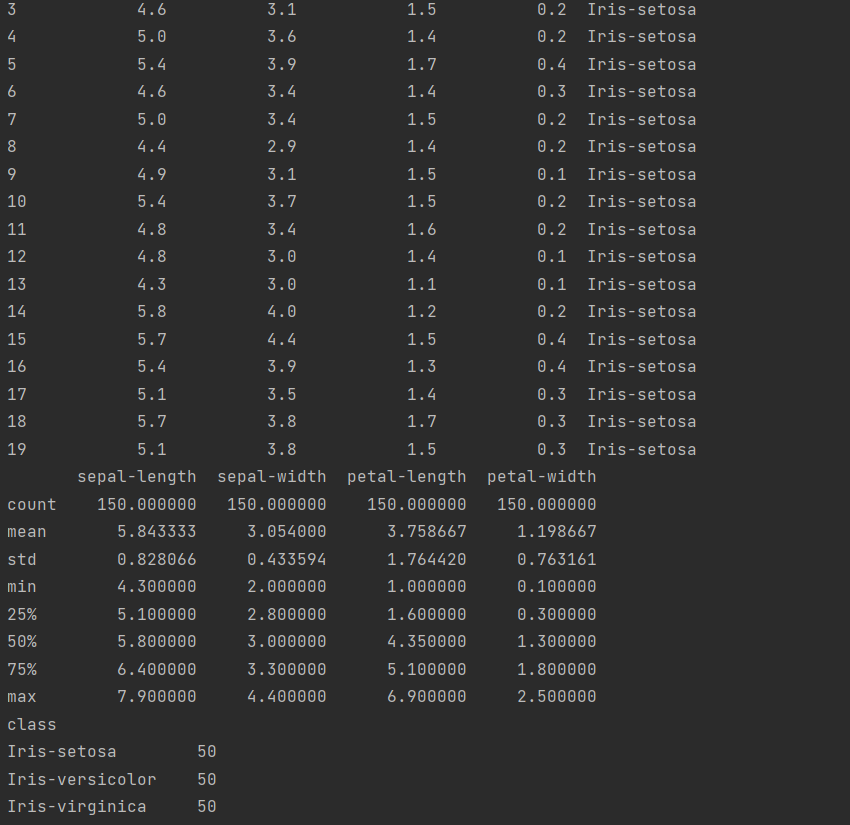
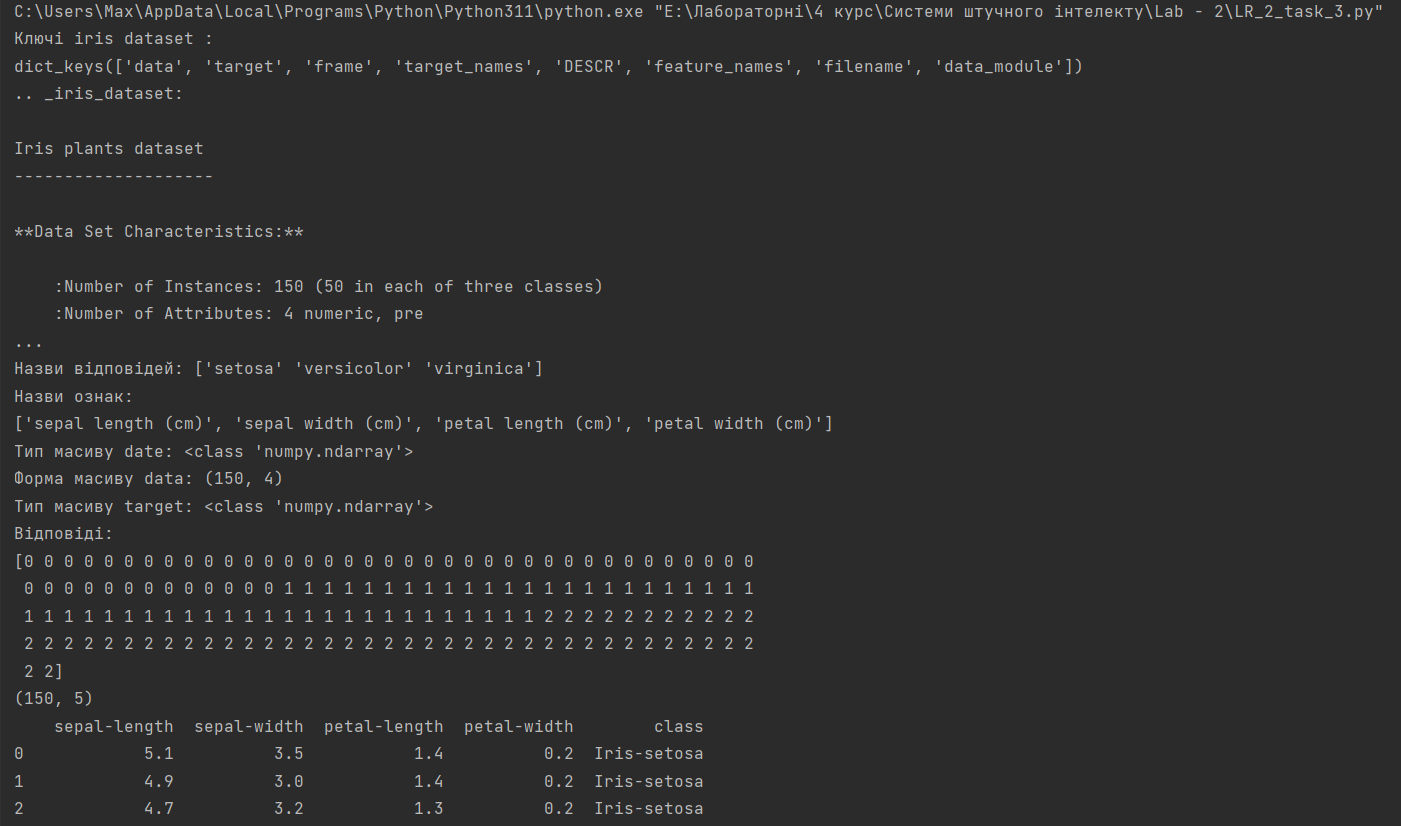


Рис. 9 Результат виконання програми

Квітка належала до класу Iris-setosa.

З діаграм можемо зробити висновок, що найкраще показала себе модель лінійного дискримінантного аналізу.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг програми:

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_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  
  
input\_file = "income\_data.txt"  
X = []  
Y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 2500  
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  
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:  
 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 = LogisticRegression(solver='liblinear', multi\_class='ovr')  
#classifier = LinearDiscriminantAnalysis()  
#classifier = KNeighborsClassifier()  
#classifier = DecisionTreeClassifier()  
#classifier = GaussianNB()  
classifier = SVC(gamma='auto')  
  
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)) + "%")  
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)) + "%")  
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 = 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]  
predicate\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicate\_class)[0])

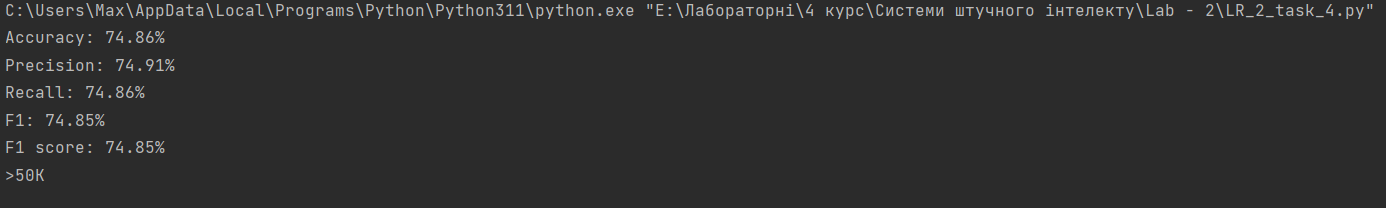


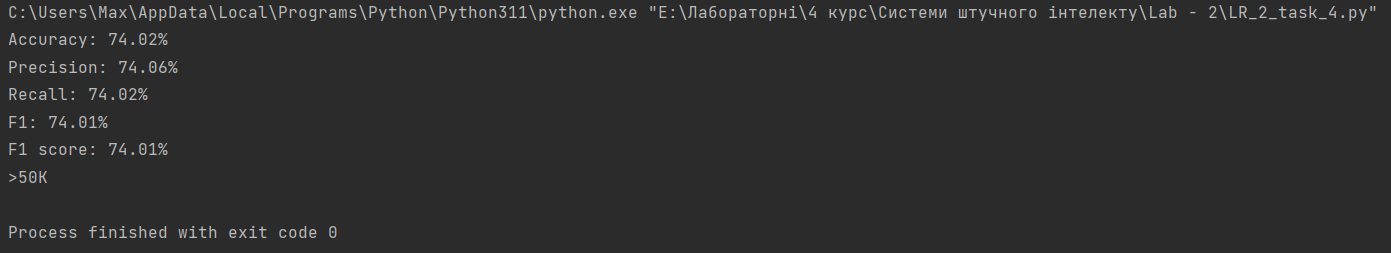
Рис.10 Точність класифікатора LR

Рис. 11 Точність класифікатора LDA

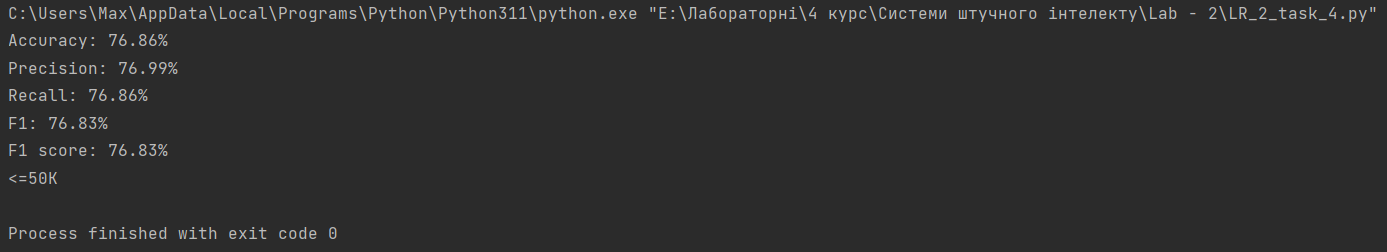


Рис. 12 Точність класифікатора KNN

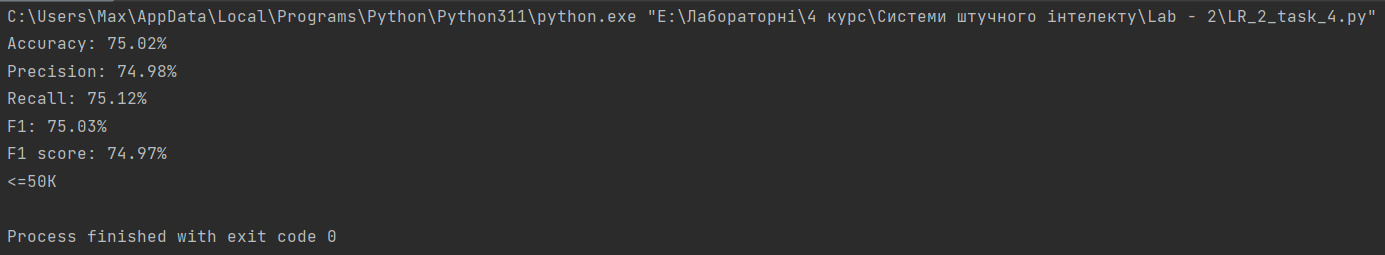


Рис. 13 Точність класифікатора CART

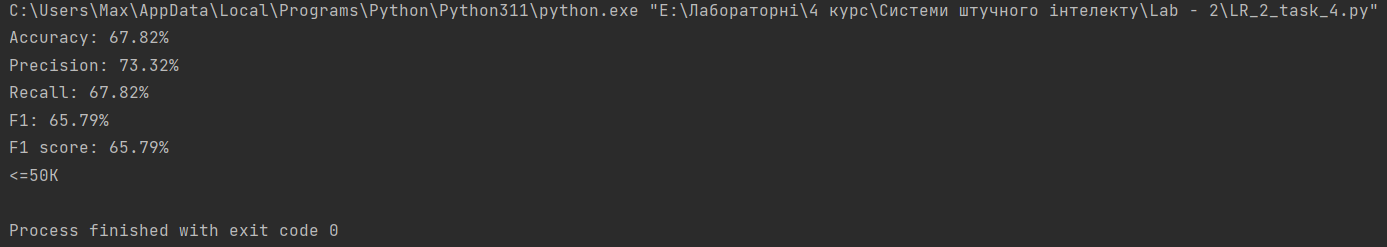


Рис. 14 Точність класифікатора NB

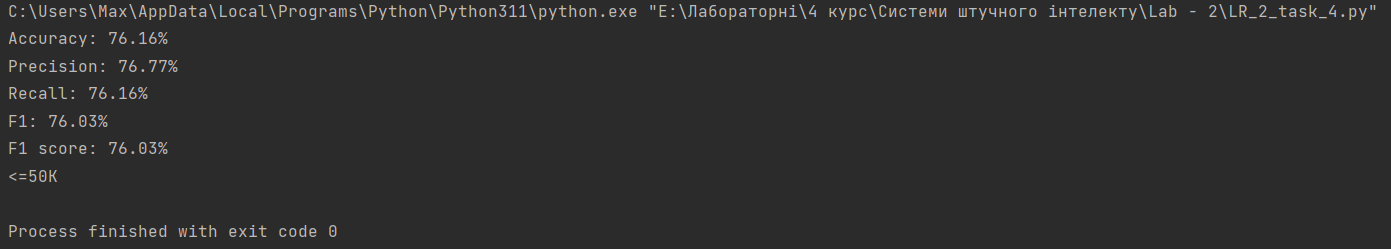


Рис. 15 Точність класифікатора SVM

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг програми:

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")

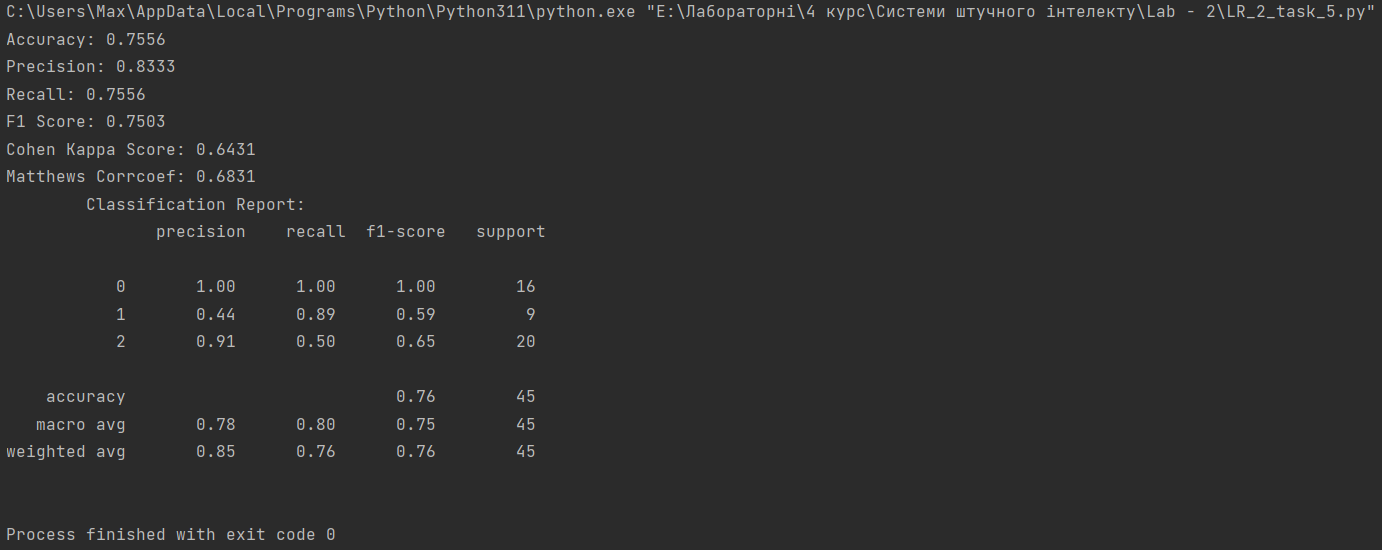


Рис. 16 Результат виконання

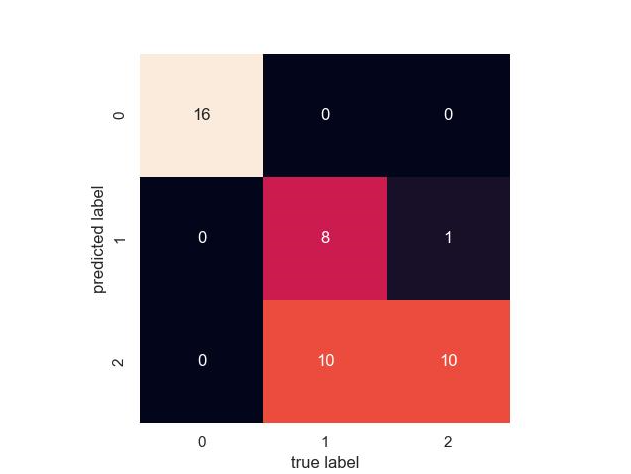


Рис. 16 Матриця невідповідності

З результату бачимо, що отримано r1, recall, коеф. Коена Каппа – це статичні значення, що вимірює міжрегіональну згоду на категоріальні предметі і вважається більш надійнішим аніж розрахунок у відсотках. Також було отримано коефіцієнт кореляції Метьюза – використовується в машинному навчанні, як міра якості бінарних мультикласних класифікацій.

Матриця невідповідності – це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання. Кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу.

**Висновки:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.