**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:**

Лістинг коду:

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import cross\_val\_score, train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_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  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
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)  
classifier = OneVsOneClassifier(LinearSVC(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.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
precision = precision\_score(y\_test, y\_test\_pred, average='weighted')  
recall = recall\_score(y\_test, y\_test\_pred, average='weighted')  
f1 = f1\_score(y\_test, y\_test\_pred, average='weighted')  
print(f"Accuracy: {round(accuracy \* 100, 2)}%")  
print(f"Precision: {round(precision \* 100, 2)}%")  
print(f"Recall: {round(recall \* 100, 2)}%")  
print(f"F1 score: {round(f1 \* 100, 2)}%")  
f1\_cross\_val = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1 score (cross-validation): " + str(round(100 \* f1\_cross\_val.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:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded)  
predicted\_class = classifier.predict(input\_data\_encoded.reshape(1, -1))  
print("Predicted class for input data: ", label\_encoder[-1].inverse\_transform(predicted\_class)[0])

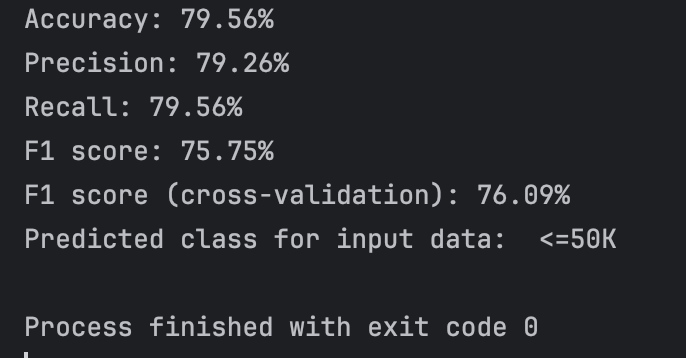


Рис. 1. Результат виконання програми

Вік – числова, робочий клас – категоріальна, fnlwgt – вага вибірки – числова, освіта – категоріальна, education-num – найвищий рівень освіти – числова, сімейний стан – категоріальна, сфера роботи – категоріальна, взаємовідносини – категоріальна, раса – категоріальна, стать – категоріальна, приріст капіталу – числова, збиток капіталу – числова, годин на тиждень – числова, рідна країна –категоріальна.

Тестова точка належить до класу <=50K.

**Завдання 2:**

Поліноміальне ядро.

Лістинг коду:

import numpy as np  
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 = 'income\_data.txt'  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 1000  
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)  
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8))  
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(SVC(kernel='poly', degree=8))  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
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:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])  
num\_folds = 3  
accuracy\_values = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=num\_folds)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=num\_folds)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=num\_folds)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")

A screen shot of a computer

Description automatically generated

Рис. 2. Результат виконання програми

Кількість точок для цього алгоритму було зменшено до тисячі, щоб отримати принаймні якийсь результат, оскільки він дуже вимогливий до апаратних ресурсів. Зі зменшенням кількості точок також знижуються і показники метрик.

Гаусове ядро.

Лістинг коду:

import numpy as np  
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 = '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)  
classifier = OneVsOneClassifier(SVC(kernel='rbf'))  
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(SVC(kernel='rbf'))  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
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:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])  
num\_folds = 3  
accuracy\_values = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=num\_folds)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=num\_folds)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=num\_folds)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")

A screen shot of a computer

Description automatically generated

Рис. 3. Результат виконання програми

Сигмоїдальне ядро.

Лістинг коду:

import numpy as np  
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 = '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)  
classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))  
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(SVC(kernel='sigmoid'))  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
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:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])  
num\_folds = 3  
accuracy\_values = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=num\_folds)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=num\_folds)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=num\_folds)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")

A screen shot of a computer

Description automatically generated

Рис. 4. Результат виконання програми

Згідно з результатами тренувань, гаусове ядро найкраще справляється із класифікацією для цього завдання. Можливо, поліноміальне ядро показало б кращі результати на 25000 точках, але обмеження швидкодії алгоритму не дозволяють перевірити це на практиці.

**Завдання 3:**

Лістинг коду:

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

A screenshot of a computer code

Description automatically generated

Рис. 5. Результат виконання програми

Лістинг коду:

from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
import matplotlib  
matplotlib.use('TkAgg')  
from matplotlib import pyplot  
  
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()

A screenshot of a computer

Description automatically generated

Рис. 6. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 7. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 8. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 9. Результат виконання програми

Лістинг коду:

from sklearn.svm import SVC  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import classification\_report  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
import numpy as np  
  
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 = [('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')), ('LDA', LinearDiscriminantAnalysis()),  
 ('KNN', KNeighborsClassifier()), ('CART', DecisionTreeClassifier()), ('NB', GaussianNB()),  
 ('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))  
knn = KNeighborsClassifier(n\_neighbors=1)  
knn.fit(X\_train, Y\_train)  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("Форма масива X\_new: {}".format(X\_new.shape))  
prediction = knn.predict(X\_new)  
print("Прогноз: {}".format(prediction))  
print("Оцінка тестового набору: {:.2f}".format(knn.score(X\_validation, Y\_validation)))

A black background with white numbers

Description automatically generated

Рис. 10. Результат виконання програми

A black screen with white text

Description automatically generated

Рис. 11. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 12. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 13. Результат виконання програми

**Завдання 4:**

Лістинг коду:

from pandas import read\_csv  
import matplotlib  
import numpy as np  
from sklearn import preprocessing  
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  
  
matplotlib.use('TkAgg')  
  
input\_file = 'income\_data.txt'  
dataset = read\_csv(input\_file)  
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)  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, y, test\_size=0.20, random\_state=1)  
models = [('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')), ('LDA', LinearDiscriminantAnalysis()),  
 ('KNN', KNeighborsClassifier()), ('CART', DecisionTreeClassifier()), ('NB', GaussianNB()),  
 ('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()

A black background with white numbers

Description automatically generated

Рис. 14. Результат виконання програми

A graph of a graph with a number of boxes

Description automatically generated with medium confidence

Рис. 15. Результат виконання програми

Метод класифікації LDA є найкращим для розв'язання цього завдання, оскільки має найвищу метрику точності (accuracy) і найменше стандартне відхилення.

**Завдання 5:**

Лістинг коду:

import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import RidgeClassifier  
from sklearn import metrics  
import matplotlib.pyplot as plt  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO  
import seaborn as sns  
  
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))  
sns.set()  
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")  
f = BytesIO()  
plt.savefig(f, format="svg")

A diagram of different colored squares

Description automatically generated

Рис. 16. Результат виконання програми

A screenshot of a computer

Description automatically generated

Рис. 17. Результат виконання програми

Налаштування класифікатора Ridge:

tol — параметр точності,

solver — розв'язувач для виконання обчислювальних процедур (в даному випадку застосовується стохастичний середній градієнт).

Використовувані показники якості:

* Точність ≈ 76%,
* Прецизійність ≈ 83%,
* Чутливість ≈ 76%,
* F1-оцінка ≈ 76%,
* Коефіцієнт Каппа Коена ≈ 64%,
* Коефіцієнт кореляції Метьюза ≈ 68%.

Зображення “Confusion.jpg” показує дані у вигляді квадратної матриці з кольоровим відображенням.

Коефіцієнт Каппа Коена вимірює ефективність моделей машинного навчання.

Коефіцієнт кореляції Метьюза — це міра якості бінарної класифікації, яка залишається надійною навіть за нерівномірного розподілу класів.

Посилання на репозиторій на GitHub: <https://github.com/vladyslavgeyna/artificial-intelligence-systems/tree/main/lab2>.

***Висновки:*** в ході виконання лабораторної роботи ми,використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчилися їх порівнювати.