**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:**

**Завдання 2.1**. Класифікація за допомогою машин опорних векторів (SVM).

***Випишіть у звіт всі 14 ознак з набору даних – їх назви та що вони позначають та вид (числові чи категоріальні).***

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Назва** | **Позначають** | **Вид** |
| 1 | Age | Вік | Integer (Числові) |
| 2 | Workclass | Дохід | Categorical (Категоріальні) |
| 3 | Fnlwgt | - | Integer (Числові) |
| 4 | Education | Освіта | Categorical (Категоріальні) |
| 5 | Education-num | Рівень освіти | Integer (Числові) |
| 6 | Marital-status | Сімейний стан | Categorical (Категоріальні) |
| 7 | Occupation | Професія | Categorical (Категоріальні) |
| 8 | Relationship | Відносини | Categorical (Категоріальні) |
| 9 | Race | Раса | Categorical (Категоріальні) |
| 10 | Sex | Стать | Binary (Категоріальні) |
| 11 | Capital-gain | Приріст капіталу | Integer (Числові) |
| 12 | Capital-loss | Збиток капіталу | Integer (Числові) |
| 13 | Hours-per-week | Години на тиждень | Integer (Числові) |
| 14 | Native-country | Батьківщина | Categorical (Категоріальні) |
| 15 | Income | Дохід | Binary (Числові) |

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, cross\_val\_score  
  
# Вхідний файл, який містить дані  
input\_file = 'income\_data.txt'  
  
# Читання даних  
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  
  
# Перетворення на масив numpy  
X = np.array(X)  
  
# Перетворення рядкових даних на числові  
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\_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, max\_iter=10000))  
classifier.fit(X\_train, y\_train)  
  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_macro', cv=3)  
print("F1 score: " + str(round(100\*f1.mean(), 2)) + "%")  
  
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)) + "%")  
  
# Передбачення результату для тестової точки даних  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male',  
'0', '0', '40', 'United-States']  
  
# Кодування тестової точки даних  
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)  
  
# Використання класифікатора для кодованої точки даних  
# та виведення результату  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат виконання програми:

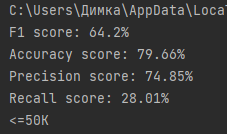


Рис. 2.1.1 – Результат виконання завдання.

***Висновок*:**

Ця тестова точка відноситься до групи людей, заробляння яких становить менше або дорівнює 50 тисячам.

**Завдання 2.2**. Порівняння якості класифікаторів SVM з нелінійними ядрами

Лістинг програми (з поліноміальним ядром):

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
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  
  
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\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
  
classifier = SVC(kernel="poly", degree=8)  
classifier.fit(X\_train\_scaled, y\_train)  
y\_test\_pred = classifier.predict(X\_test\_scaled)  
  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
precision = precision\_score(y\_test, y\_test\_pred)  
recall = recall\_score(y\_test, y\_test\_pred)  
f1 = f1\_score(y\_test, y\_test\_pred)  
  
print("Accuracy score: {:.2f}%".format(100 \* accuracy))  
print("Precision score: {:.2f}%".format(100 \* precision))  
print("Recall score: {:.2f}%".format(100 \* recall))  
print("F1 score: {:.2f}%".format(100 \* f1))

Результат виконання програми:

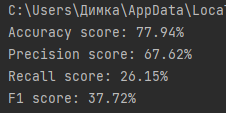


Рис. 2.2.1 – Результат виконання завдання (1).

Лістинг програми (з гаусовим ядром):

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
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  
  
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\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
  
classifier = SVC(kernel="rbf")  
classifier.fit(X\_train\_scaled, y\_train)  
y\_test\_pred = classifier.predict(X\_test\_scaled)  
  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
precision = precision\_score(y\_test, y\_test\_pred)  
recall = recall\_score(y\_test, y\_test\_pred)  
f1 = f1\_score(y\_test, y\_test\_pred)  
  
print("Accuracy score: {:.2f}%".format(100 \* accuracy))  
print("Precision score: {:.2f}%".format(100 \* precision))  
print("Recall score: {:.2f}%".format(100 \* recall))  
print("F1 score: {:.2f}%".format(100 \* f1))

Результат виконання програми:

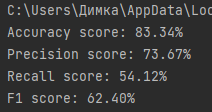


Рис. 2.2.2 – Результат виконання завдання (2).

Лістинг програми (з сигмоїдальним ядром):

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
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  
  
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\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
  
classifier = SVC(kernel="sigmoid")  
classifier.fit(X\_train\_scaled, y\_train)  
y\_test\_pred = classifier.predict(X\_test\_scaled)  
  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
precision = precision\_score(y\_test, y\_test\_pred)  
recall = recall\_score(y\_test, y\_test\_pred)  
f1 = f1\_score(y\_test, y\_test\_pred)  
  
print("Accuracy score: {:.2f}%".format(100 \* accuracy))  
print("Precision score: {:.2f}%".format(100 \* precision))  
print("Recall score: {:.2f}%".format(100 \* recall))  
print("F1 score: {:.2f}%".format(100 \* f1))

Результат виконання програми:

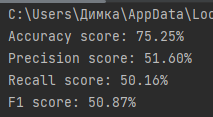


Рис. 2.2.3 – Результат виконання завдання (3).

***Висновок:***

Найбільш точним і дбайливим класифікатором є нелінійний SVM з гаусовим ядром, але що стосується повноти, найкращим виявився нелінійний SVM з поліноміальним ядром. Загалом, для виконання завдання найкращим класифікатором є той, що використовує гаусове ядро.

**Завдання 2.3**. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

Лістинг програми:

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("Значення ознак для п'яти прикладів: {}".format(iris\_dataset['data'][:5]))  
print("Тип масиву target: {}".format(type(iris\_dataset['target'])))  
print("Відповіді:\n{}".format(iris\_dataset['target']))

Результат виконання програми:

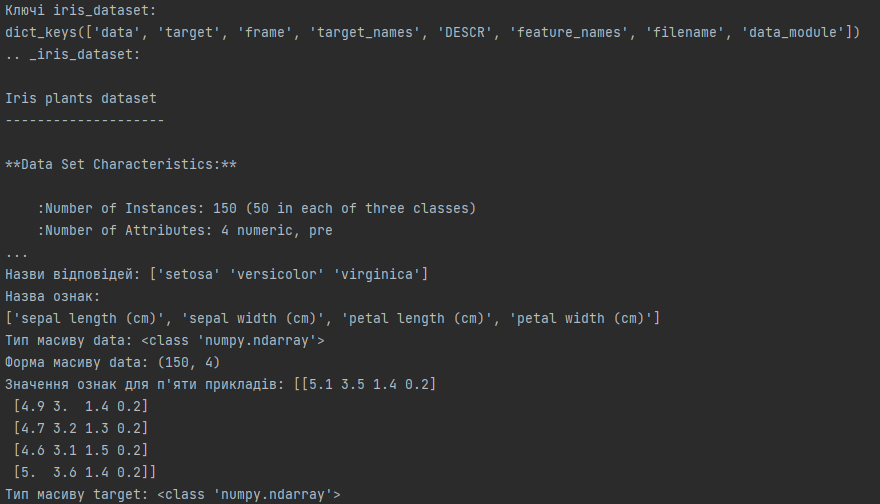


Рис. 2.3.1 – Результат виконання завдання (1).

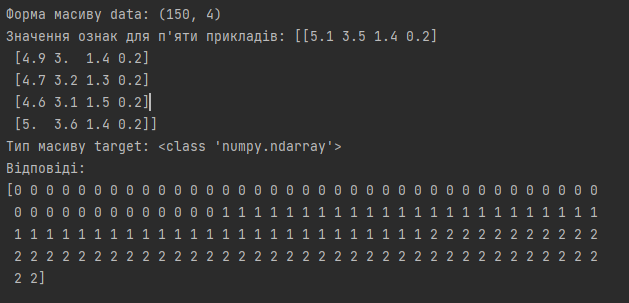


Рис. 2.3.2 – Результат виконання завдання (2).

Лістинг програми:

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]))

Результат виконання програми:

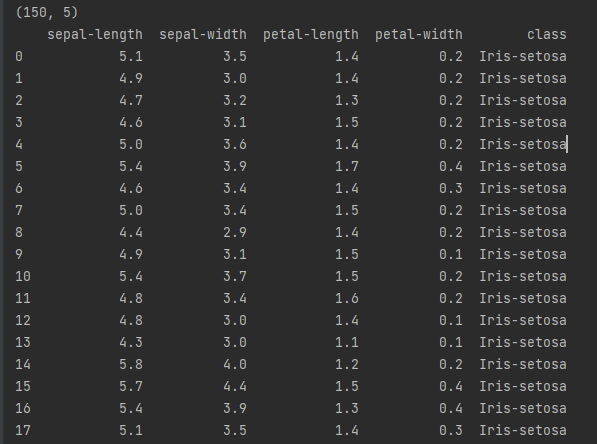


Рис. 2.3.3 – Результат виконання завдання (1).

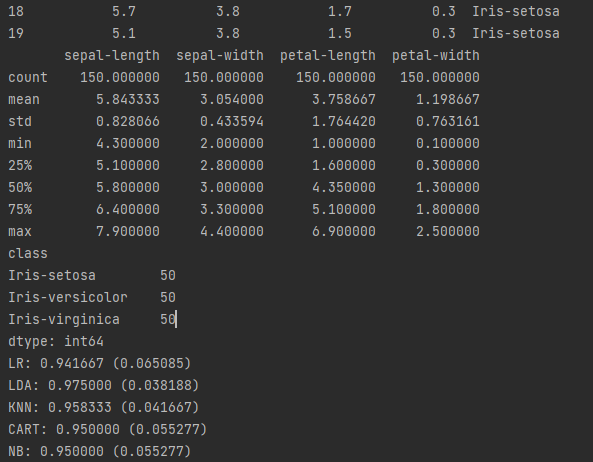


Рис. 2.3.4 – Результат виконання завдання (2).

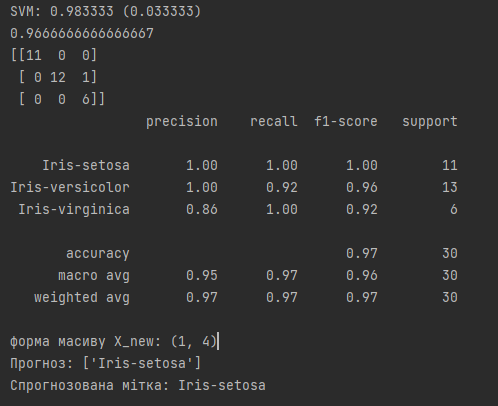


Рис. 2.3.5 – Результат виконання завдання (3).

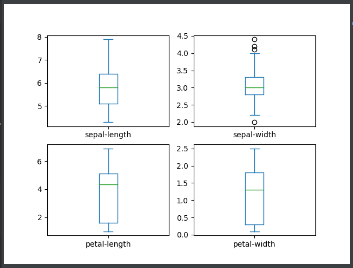


Рис. 2.3.6 – Результат виконання завдання (3).

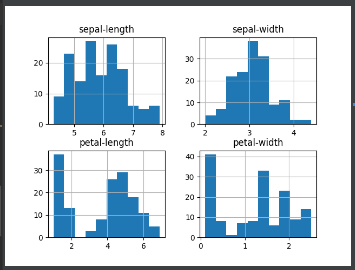


Рис. 2.3.7 – Результат виконання завдання (4).

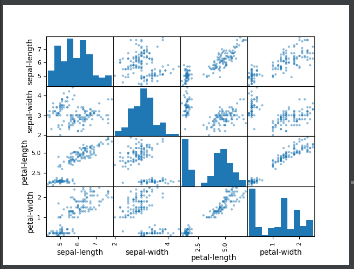


Рис. 2.3.8 – Результат виконання завдання (5).

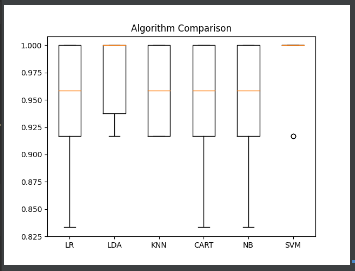


Рис. 2.3.9 – Результат виконання завдання (6).

***Отримані графіки та результати занесіть у звіт Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим.***

* *Обраний метод класифікації - "SVM" (Support Vector Machine) - вважаю найкращим через його здатність працювати добре навіть у випадку складних нелінійних вибірок із великою кількістю ознак. В даному випадку, модель "SVC" (SVM з ядром "auto") показала високу точність під час перевірки на тестовій вибірці та має потенціал для роботи з новими даними.*

***Висновок:***

Квітка належить до класу Setosa. Вдалося досягти 0.966…7 показника якості.

**Завдання 2.4**. Порівняння якості класифікаторів для набору даних завдання 2.1.

Лістинг програми:

from sklearn import preprocessing  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score, 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'  
max\_datapoints = 25000  
  
X = []  
y = []  
  
count\_class1 = 0  
count\_class2 = 0  
  
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.strip().split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
 elif 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=0.2, 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()))

Результат виконання програми:

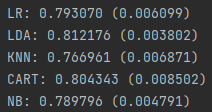


Рис. 2.4.1 – Результат виконання завдання.

**Завдання 2.5**. Класифікація даних лінійним класифікатором Ridge.

Лістинг програми:

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

Результат виконання програми:

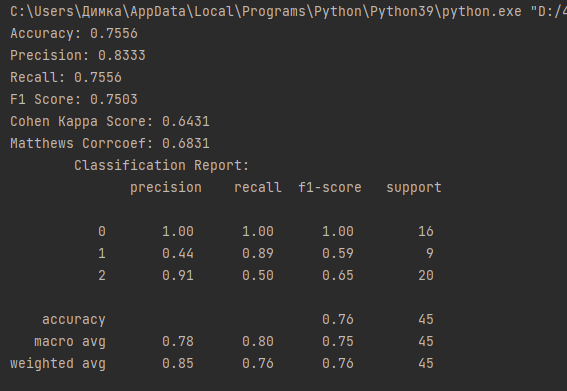


Рис. 2.5.1 – Результат виконання завдання (1).

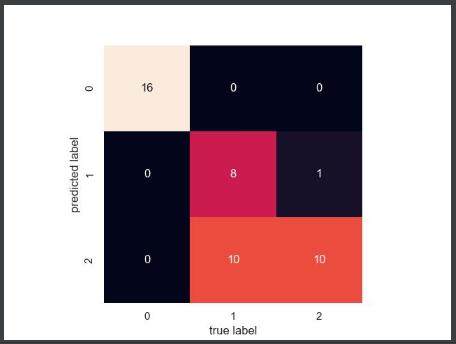


Рис. 2.5.2 – Результат виконання завдання (2).

***Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають.***

* *В класифікатора Ridge використовані наступні налаштування: параметр tol встановлено на значення 1e-2 (це допуск для зупинки оптимізаційного алгоритму), і як рішальник використовується "sag" (Stochastic Average Gradient Descent), що є одним із методів оптимізації для RidgeClassifier.*

***Опишіть які показники якості використовуються та їх отримані результати. Вставте у звіт та поясніть зображення Confusion.jpg***

* *Accuracy, Precision, Recall, F1 score, Cohen Kappa Score, Matthews Correlation Coefficient.*
* *Зображення "Confusion.jpg" відображає матрицю помилок, де по діагоналі показано правильно класифіковані приклади для кожного класу, а поза діагоналлю - помилкові класифікації.*

***Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.***

* *Коефіцієнт Коена Каппа і коефіцієнт кореляції Метьюза використовуються для вимірювання узгодженості між фактичними та передбаченими класами. Коефіцієнт Коена Каппа враховує випадковість та відображає, наскільки збігаються фактичні та передбачені класи, враховуючи інтеркориговану випадковість. Коефіцієнт кореляції Метьюза також враховує випадковість, і він вимірює ступінь кореляції між спостереженнями та передбаченнями, де 1 вказує на ідеальну узгодженість, -1 на повну протилежність, а 0 - на випадкову узгодженість. Ці коефіцієнти оцінюють ступінь надійності класифікатора.*

***Посилання на репозиторій:*** [**https://github.com/GrunytskyDmytro/Lab2\_AI**](https://github.com/GrunytskyDmytro/Lab2_AI)

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