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:

Ознака	Можливі значення
age	continuous
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
fnlwgt	continuous
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, As□soc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
education-num	continuous
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	Tech-support, Craft-repair, Other-service, Sales, Execmanagerial, Prof-specialty, Handlers-cleaners, Machine□op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	Female, Male
capital-gain	continuous
capital-loss	continuous
hours-per-week	continuous
native-country	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

Напишемо код для класифікації:

```
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
from sklearn.model_selection import 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)
# Create SVM classifier
classifier = OneVsOneClassifier(LinearSVC(random_state=0, dual=False))
# Train the classifier
classifier.fit(X, 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, dual=False))
classifier.fit(X_train, 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)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.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 = [-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]]))
    count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

# Run classifier on encoded datapoint and print output
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Результат:

```
F1: 76.01%
Акуратність: 79.66%
Повнота: 79.66%
Точність: 78.88%
<=50K
```

Рис. 1. Результати класифікатора щорічного прибутку

Задана точка відноситься до класифікатора «<=50K»

Завдання №2:

Код:

```
import numpy as np
import matplotlib.pyplot as plt
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 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)
# Create SVM classifier
classifier = OneVsOneClassifier(SVC(kernel='sigmoid')) // kernel = 'poly' kernel
= 'rbf'
# Train the classifier
classifier.fit(X, 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(SVC(kernel='sigmoid')) // kernel = 'poly' kernel
= 'rbf'
classifier.fit(X_train, 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)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.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', 'UnitedStates']
# Encode test datapoint
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]]))
    count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
# Run classifier on encoded datapoint and print output
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Результат:

```
F1: 71.95%
Акуратність: 78.61%
Повнота: 78.61%
Точність: 83.06%
<=50K
```

Рис. 2. Результати SVM з гаусовим ядром

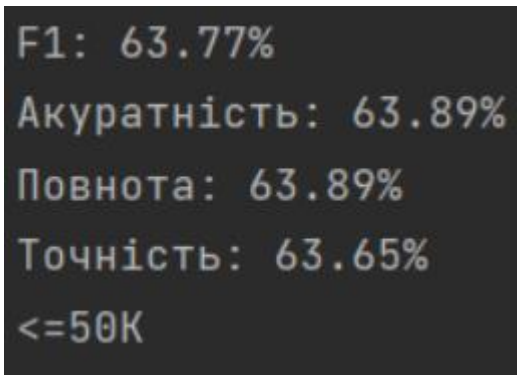

Рис. 3. Результати SVM з сигмоїдальним ядром

Завдання №3:

Ознайомлення зі структурою даних та результати:

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Тип масиву target: <class 'numpy.ndarray'>
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Рис. 4. Ознайомлення зі структурою даних та результати

Код візуалізації:

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
# Load dataset
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)
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(
    2, 2), sharex=False, sharey=False)
```

```
pyplot.show()
# histograms
dataset.hist()
pyplot.show()
# scatter plot matrix
scatter_matrix(dataset)
pyplot.show()
```

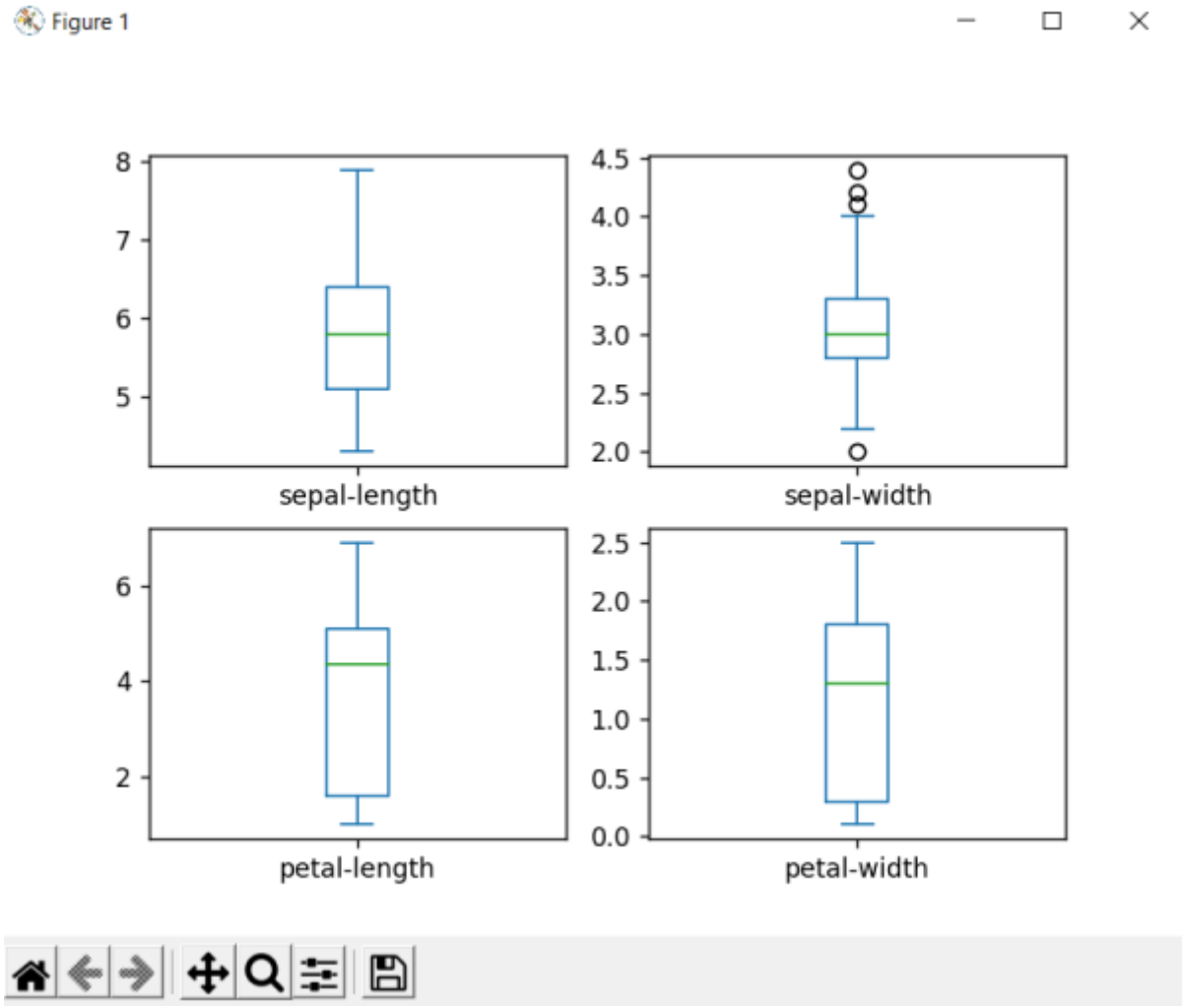

Рис. 5. Одновимірний графік

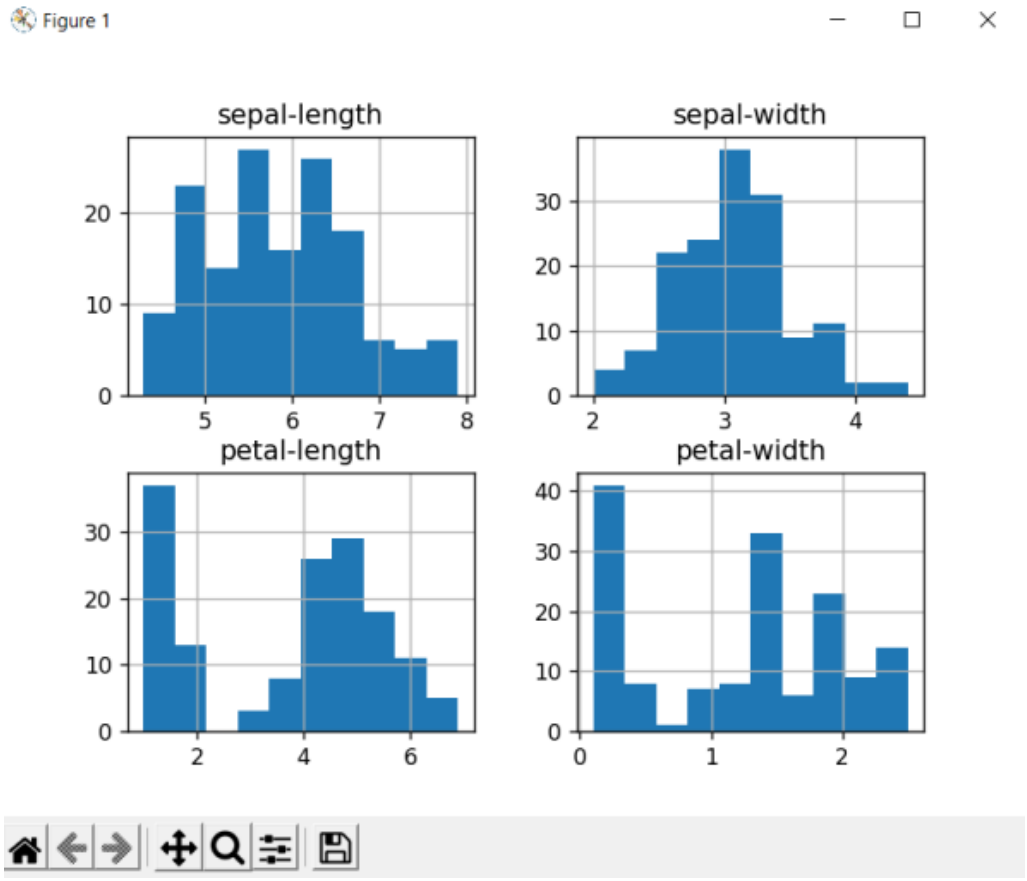

Рис. 6. Гістограма розподілу атрибутів датасета

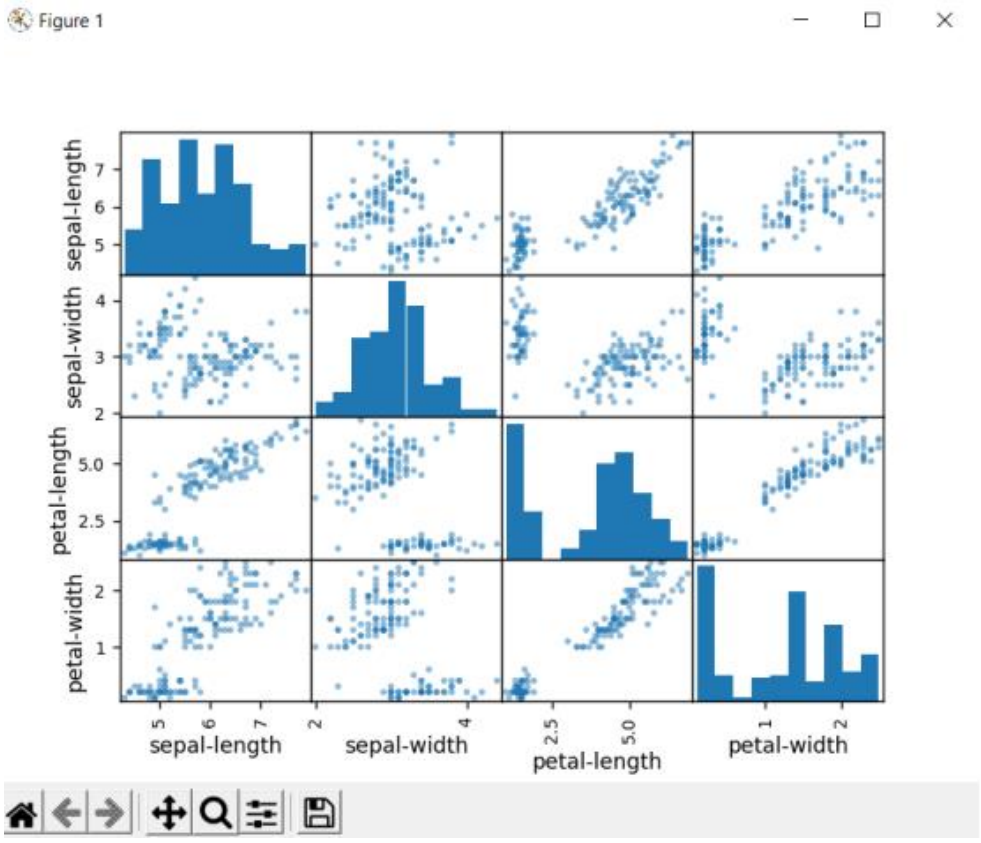

Рис. 7. Багатовимірний графік

Протестуємо 6 різних алгоритмів:

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.038188)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

Рис. 8. Багатовимірний графік

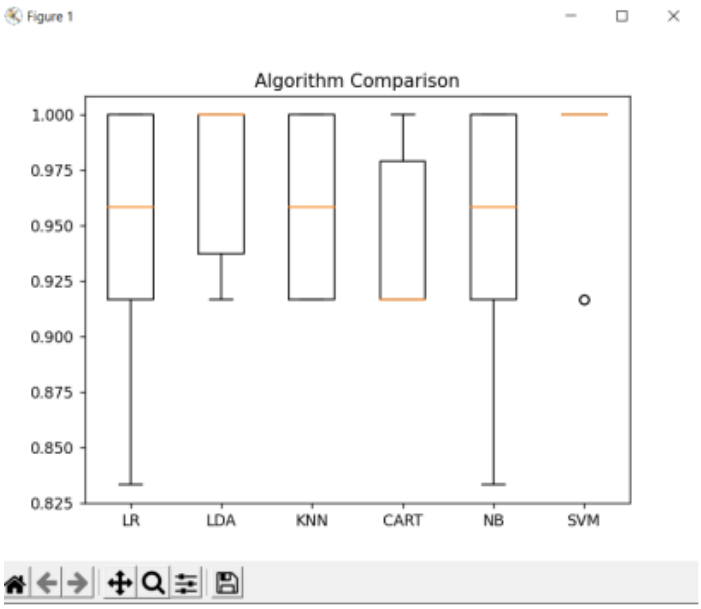

Рис. 9. Багатовимірний графік

З рисунків бачимо, що найбільш точним алгоритмом у цьому випадку є SVM. Написаний код для здійснення прогнозу:

```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))
prediction = model.predict(X_new)
print("Prediction of Species: {}".format(prediction))
```

Отримуємо:

```
X_new.shape: (1, 4)
Prediction of Species: ['Iris-setosa']
```

Рис. 10. Отриманий клас

Отже, спрогнозований сорт ірису – Iris-setosa.

Завдання №4:

Порівняймо алгоритми з минулого завдання, використовуючи задачу з першого завдання. Код:

```
import numpy as np
import matplotlib.pyplot as plt
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
from sklearn.linear_model import LinearRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
# 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)
# Cross validation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(LinearRegression())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("LR:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")
classifier = OneVsOneClassifier(LinearDiscriminantAnalysis())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("LDA:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")
classifier = OneVsOneClassifier(KNeighborsClassifier())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("KNN:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")
classifier = OneVsOneClassifier(DecisionTreeClassifier())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
```

```
print("CART:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")
classifier = OneVsOneClassifier(SVC())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("SVM:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(
    classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")
X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5], [6, 5], [5.6, 5], [3.3,
                                                                              0.4]
, [3.9, 0.9], [2.8, 1], [0.5, 3.4], [1, 4], [0.6, 4.9]])
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])
# Create the logistic regression classifier
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)
# classifier = linear_model.LogisticRegression(solver='liblinear', C=100)
# Train the classifier
classifier.fit(X, y)
# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)
```

Результат:

```
LR:
F1: 31.36%
Акуратність: 36.61%
Повнота: 36.61%
Точність: 81.24%

LDA:
F1: 79.35%
Акуратність: 81.14%
Повнота: 81.14%
Точність: 79.86%

KNN:
F1: 48.04%
Акуратність: 46.48%
Повнота: 46.48%
Точність: 70.48%

CART:
F1: 80.69%
Акуратність: 80.57%
Повнота: 80.7%
Точність: 80.75%

SVM:
F1: 71.95%
Акуратність: 78.61%
Повнота: 78.61%
Точність: 83.06%
```

Рис. 11. Результати порівняння алгоритмів

Отже, переглянувши всі результати, робимо висновок, що для цієї задачі найбільше підійде алгоритм CART.

Завдання №5:

Виконаймо класифікацію даних лінійним класифікатором Ridge. Код:

```
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn import metrics
import numpy as np
from sklearn.datasets import load_iris
```

```
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import confusion_matrix
from io import BytesIO  # neded for plot
import seaborn as sns
sns.set()
iris = load_iris()
X, y = iris.data, iris.target
Xtrain, X_test, 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(X_test)
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")
```

Результат:

```
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831
        Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.44      0.89      0.59         9
           2       0.91      0.50      0.65        20

    accuracy                           0.76        45
   macro avg       0.78      0.80      0.75        45
weighted avg       0.85      0.76      0.76        45
```

Рис. 12. Класифікація даних лінійним класифікатором Ridge

У класифікаторі Ridge були застосовані налаштування tol та solver. Параметр tol задається для контролю точності рішення, тоді як solver використовується в алгоритмах обчислення. Для оцінювання якості класифікатора використовуються наступні критерії:

- Акуратність: відображає загальну точність класифікації.
- Точність: показує, наскільки точно модель ідентифікує позитивні випадки.
- Повнота: вимірює здатність моделі виявляти всі позитивні випадки.
- F1-показник: комбінує точність і повноту в один показник, що дає збалансовану оцінку якості.
- Коефіцієнт каппи Коена: статистичний показник, який вимірює надійність між двома оцінювачами для категорійних даних, з можливими значеннями від 0 до 1.
- Коефіцієнт кореляції Метьюза: використовується для визначення сили зв'язку між двома бінарними змінними, з можливими значеннями від 0 до

Ці метрики в сукупності дозволяють всебічно оцінити ефективність і надійність класифікатора Ridge.

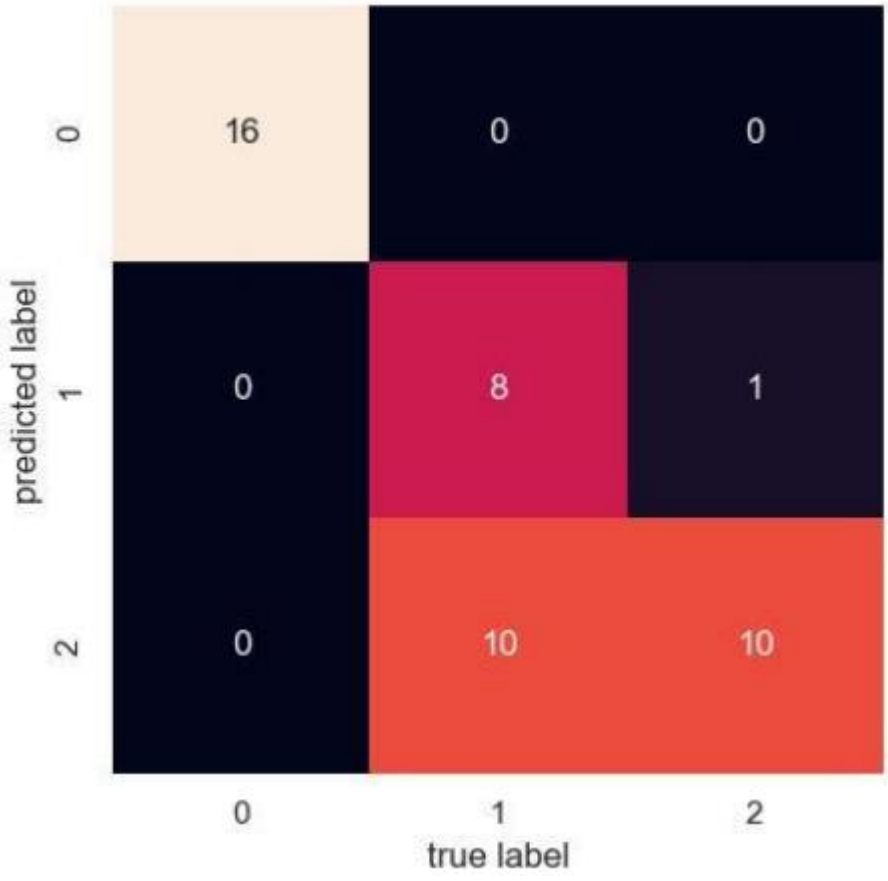

Рис. 13. Матриця з показниками якості

Висновок: на цій лабораторній роботі я, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив різні методи класифікації даних та навчився їх порівнювати.