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:

Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	684525

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:

```
PS D:\DIMA_Навчання\Системи Штучного Інтелекту\lr2>  d:; cd 'd:\DIMA_Навчання\Системи Штучного Інтелекту\lr2'; & 'C:\Python310\python.exe' 'c:\Users\korop\.vsco
de\extensions\ms-python.python-2022.6.2\pythonFiles\lib\python\debugpy\launcher' '56579' '--' 'd:\DIMA_Навчання\Системи Штучного Інтелекту\lr2\LR_2_task_3.py'
(150, 5)
     sepal-length  sepal-width  petal-length  petal-width        class
0             5.1          3.5           1.4          0.2  Iris-setosa
1             4.9          3.0           1.4          0.2  Iris-setosa
2             4.7          3.2           1.3          0.2  Iris-setosa
19            5.1          3.8           1.5          0.3  Iris-setosa
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Рис. 3.1. Структура даних

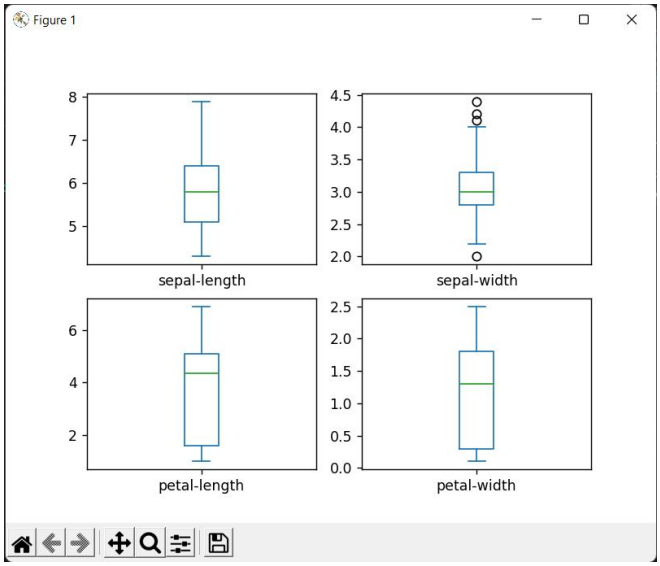

Рис. 3.2. Одновимірні графіки характеристик

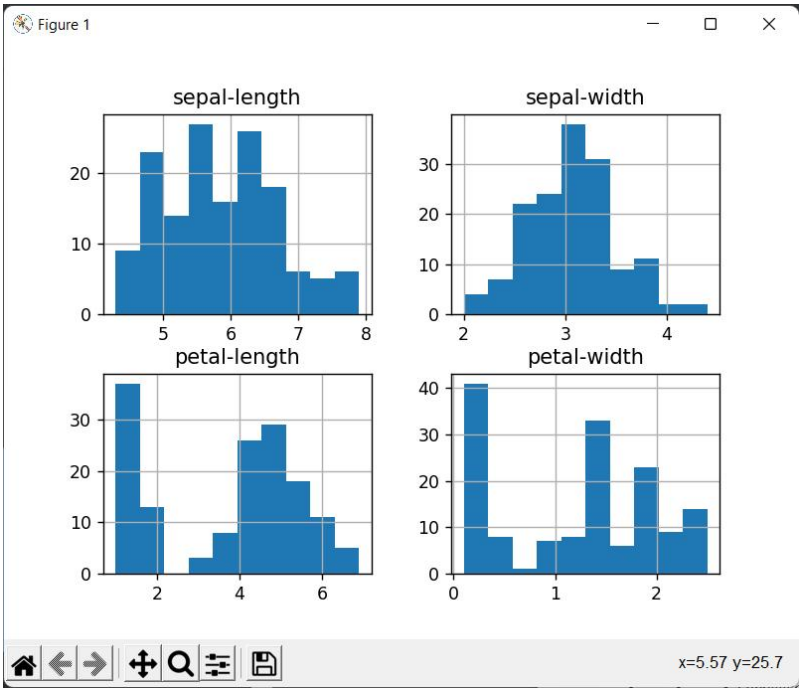

Рис. 3.3. Діаграма розмаху атрибутів

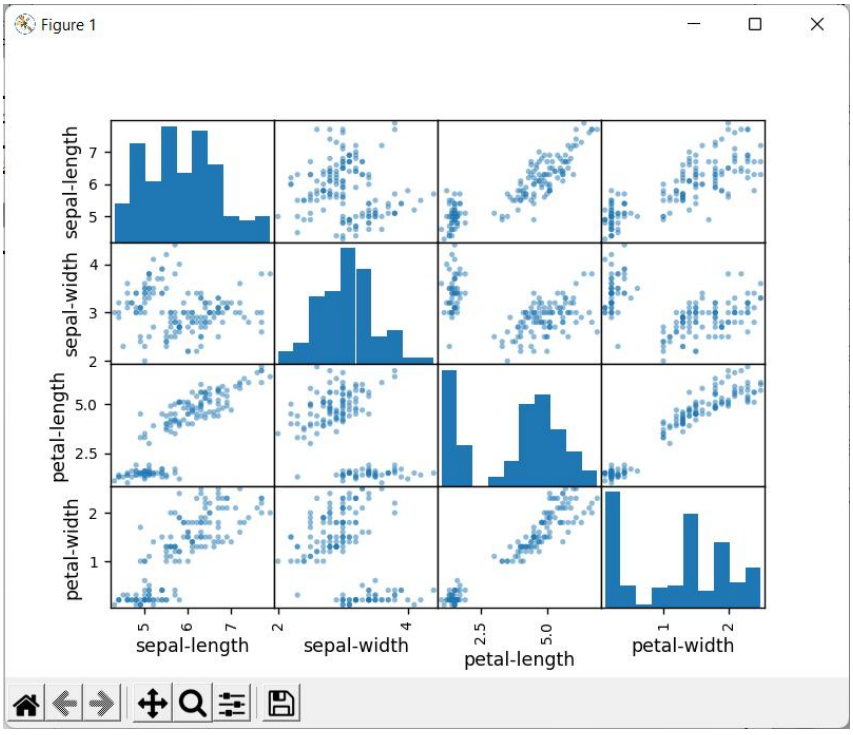

Рис. 3.4. Матриця розсіювання

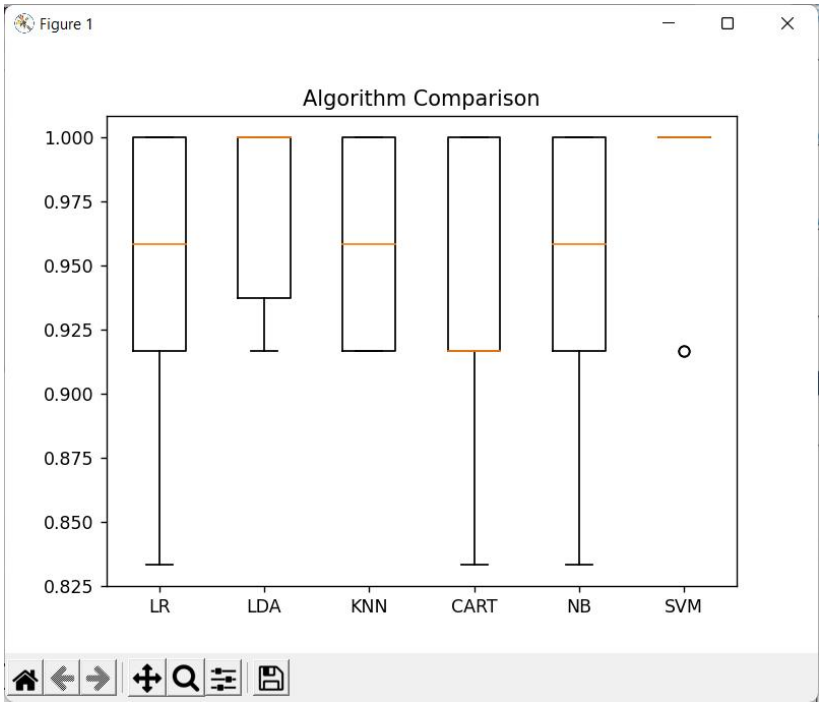

Рис. 3.5. Графік порівняння алгоритмів

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()
```

```
PS D:\DIMA_Навчання\Системи Штучного Інтелекту\lr2>
de\extensions\ms-python.python-2022.6.2\pythonFiles\l
LR: 0.791993 (0.005400)
LDA: 0.811637 (0.005701)
KNN: 0.767748 (0.003026)
CART: 0.809109 (0.009003)
NB: 0.789133 (0.006934)
```

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")
```

```
PS D:\DIMA_Навчання\Системи Штучного Інтелекту\lr2>  d:; cd 'd:
de\extensions\ms-python.python-2022.6.2\pythonFiles\lib\python\
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
```

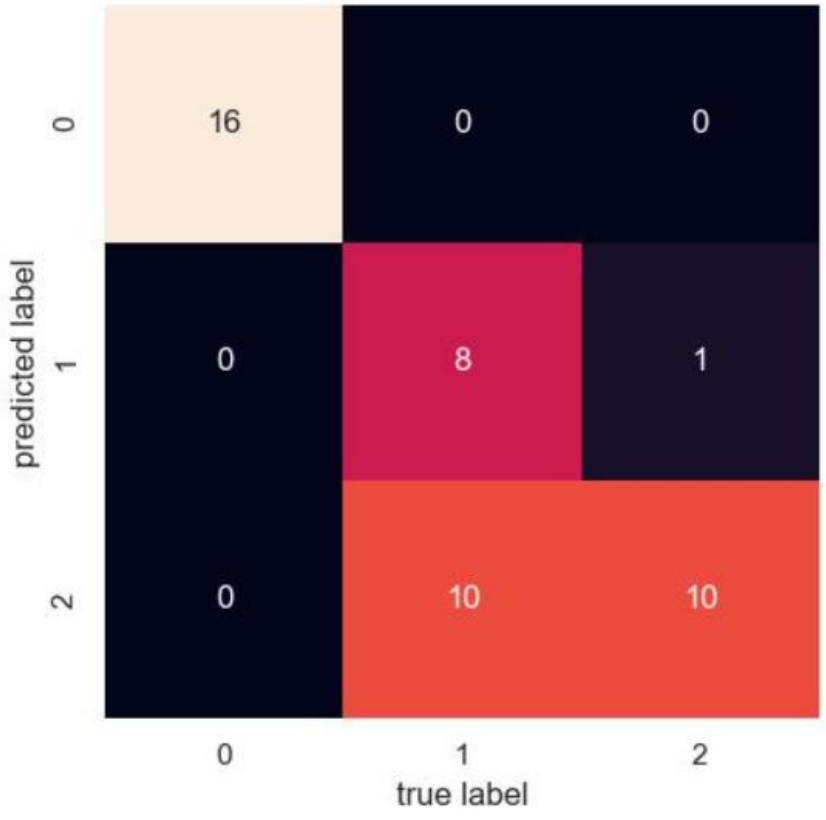

Рис. 5.1. Результат виконання

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.