Лабораторна робота №2

Системи штучного інтелекту

Виконав: ІПЗ-21-3 Осипчук Антон Олексійович

Github: https://github.com/AntonOsypchuk1/ai_lab/tree/main/lab2

2.1. Класифікація за допомогою машин опорних векторів (SVM)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_score

# Load data
input_file = './data/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))

# Cross validation
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)

# Compute F1 score of the SVM classifier
f1 = cross_val_score(classifier, X, y, scoring="f1_weighted")
print("F1 score: " + str(round(100 * f1.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]])[0])
        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])

accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:" + str(round(100 * accuracy, 2)) + "%")

precision = precision_score(y_test, y_test_pred, average="weighted")
print("Precision:" + str(round(100 * precision, 2)) + "%")

recall = recall_score(y_test, y_test_pred, average="weighted")
print("Recall:" + str(round(100 * recall, 2)) + "%")
```

Результат виконання:

```
(venv) antonosypchuk@MacBoo
F1 score: 76.07%
<=50K
Accuracy:79.56%
Precision:79.26%
Recall:79.56%
```

2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

Поліноміальне (cannot run):

```
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8))
```

Гаусове:

```
classifier = OneVsOneClassifier(SVC(kernel='rbf', random_state=0))
```

Результат:

```
(venv) venvantonosypc
F1 score: 72.32%
<=50K
Accuracy:78.19%
Precision:82.82%
Recall:78.19%
```

Сигмоїдальне:

```
classifier = OneVsOneClassifier(SVC(kernel='sigmoid', random_state=0))
```

Результат:

```
(venv) venvantonosypchu
F1 score: 63.67%
<=50K
Accuracy:60.47%
Precision:60.64%
Recall:60.47%
```

З поданих результатів можемо зробити висновок, що класифікатор з гаусовим ядром має кращу точність ніж з лінійним або ж сигмоїдальним, тому для кращої точності це ядро стане кращим вибором.

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

Результат виконання:

```
(venv) venvantonosypchuk@MacBook-Air-Anton lab2 % python3 lr_2_task3_1.py
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, predictive
...
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 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]
```

Лістинг програми:

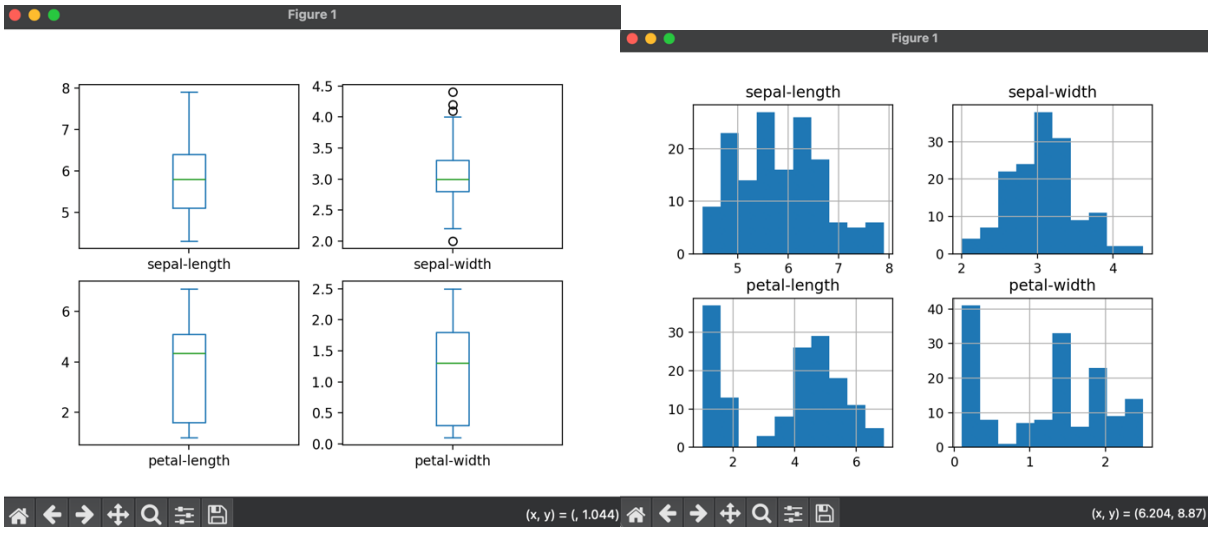
```
import numpy as np
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

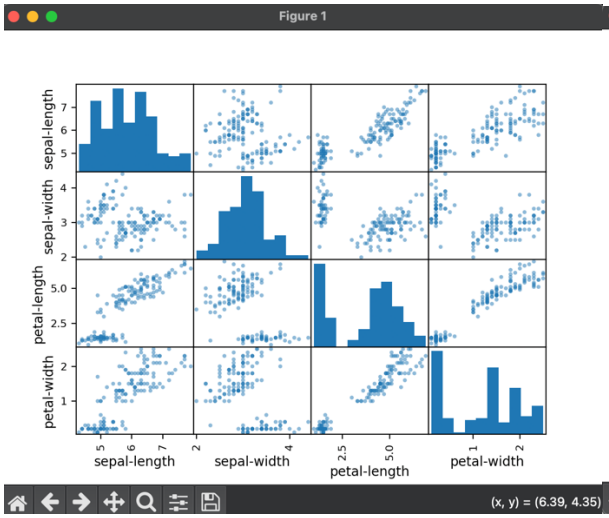
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from 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 matplotlib
matplotlib.get_backend()

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
```

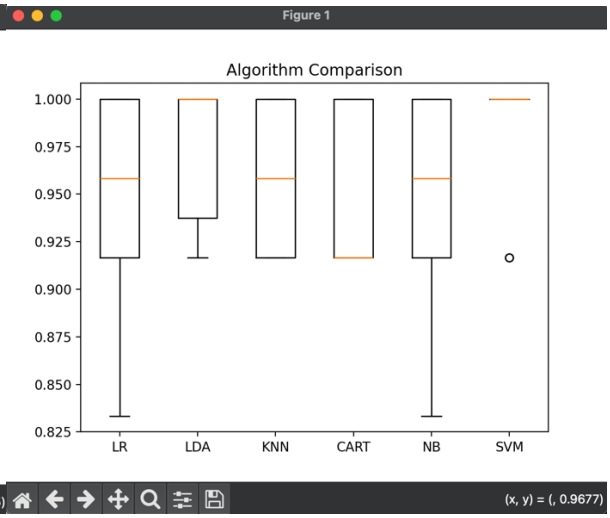
```
array = dataset.values

# 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)

# Завантажуємо алгоритми моделі
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')))

# оцінюємо модель на кожній ітерації
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))

# Завантаження даних Iris
iris = load_iris()
X, Y = iris.data, iris.target

# Розділення даних на навчальну та тестову вибірки
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)

# Новий вхідний масив
new_data = np.array([[5.0, 2.9, 1.0, 0.2]])
shape = new_data.shape
print('Форма масиву:', shape)

# Ініціалізація та навчання моделі
svc_model = SVC(gamma='auto')
svc_model.fit(X_train, Y_train)

# Прогнозування на нових даних
new_prediction = svc_model.predict(new_data)
print('Прогнозування:', new_prediction)
```

Результат виконання:








```
venvantonosypchuk@MacBook-Air-Anton lab2 % python3 lr_2_task3_2.py
(150, 5)
    sepal-length  sepal-width  ...  petal-width        class
0            5.1          3.5  ...          0.2  Iris-setosa
1            4.9          3.0  ...          0.2  Iris-setosa
2            4.7          3.2  ...          0.2  Iris-setosa
3            4.6          3.1  ...          0.2  Iris-setosa
4            5.0          3.6  ...          0.2  Iris-setosa
5            5.4          3.9  ...          0.4  Iris-setosa
6            4.6          3.4  ...          0.3  Iris-setosa
7            5.0          3.4  ...          0.2  Iris-setosa
8            4.4          2.9  ...          0.2  Iris-setosa
9            4.9          3.1  ...          0.1  Iris-setosa
10           5.4          3.7  ...          0.2  Iris-setosa
11           4.8          3.4  ...          0.2  Iris-setosa
12           4.8          3.0  ...          0.1  Iris-setosa
13           4.3          3.0  ...          0.1  Iris-setosa
14           5.8          4.0  ...          0.2  Iris-setosa
15           5.7          4.4  ...          0.4  Iris-setosa
16           5.4          3.9  ...          0.4  Iris-setosa
17           5.1          3.5  ...          0.3  Iris-setosa
18           5.7          3.8  ...          0.3  Iris-setosa
19           5.1          3.8  ...          0.3  Iris-setosa

[20 rows x 5 columns]
       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
```

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.040825)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
/Users/antonosypchuk/Desktop/lab/4 course/1 sem/AI/lab2/lr_2_task3_2.py:87: MatplotlibDeprecationWarning: Th
e 'labels' parameter of boxplot() has been renamed 'tick_labels' since Matplotlib 3.9; support for the old n
ame will be dropped in 3.11.
  pyplot.boxplot(results, labels=names)
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30

Форма масиву: (1, 4)
Прогнозування: [0]
```

Після багаторазового тестування кожного з методів класифікації, було визначено, що для розв'язання даної задачі найефективнішим є метод опорних векторів. Цей алгоритм продемонстрував найвищу середню точність серед усіх моделей, а також найменше стандартне відхилення результатів, що свідчить про його стабільність.

Під час тренування моделі з використанням цього методу вдалося досягти точності класифікації близько 96.7% (~0.967), при цьому правильно визначений клас ірису – **Iris-setosa.**

2.4. Порівняння якості класифікаторів для набору даних завдання 2.1.

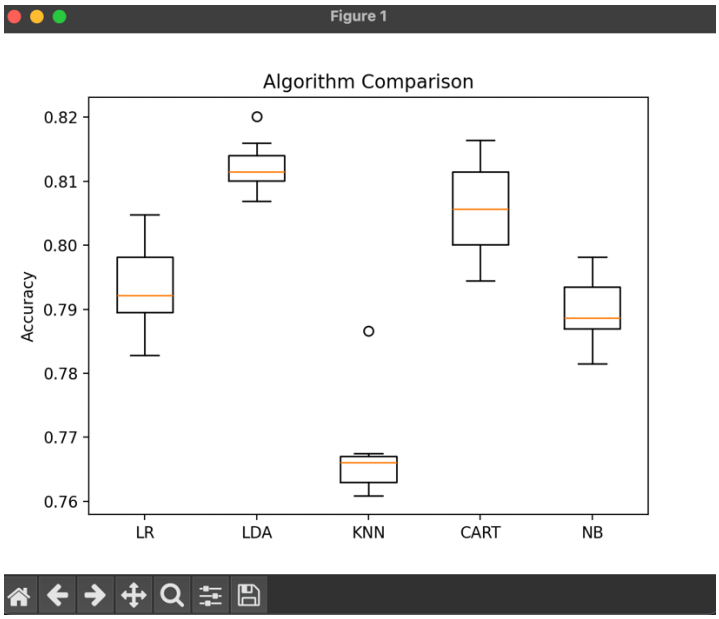
Лістинг програми:

```
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedKFold,
cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn import preprocessing

# Load data
input_file = './data/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)

# Encode categorical features
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)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)

# Define classifiers
models = [
    ('LR', LogisticRegression(solver='liblinear')),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    # ('SVM', SVC(kernel='linear', random_state=1))
]

# Evaluate each model
results = []
names = []
print("Classifier Performance:")
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(f"{name}: Mean accuracy = {cv_results.mean():.4f}, Std =
{cv_results.std():.4f}")

# Compare algorithms
import matplotlib.pyplot as plt

plt.boxplot(results, labels=names)
plt.title('Algorithm Comparison')
plt.ylabel('Accuracy')
plt.show()
```

Результат виконання:




```
(venv) venvantonosypchuk@MacBook-Air-Anton lab2 % python3 lr_2_task4.py
Classifier Performance:
LR: Mean accuracy = 0.7936, Std = 0.0065
LDA: Mean accuracy = 0.8122, Std = 0.0038
KNN: Mean accuracy = 0.7669, Std = 0.0069
CART: Mean accuracy = 0.8059, Std = 0.0068
NB: Mean accuracy = 0.7898, Std = 0.0048
/Users/antonosypchuk/Desktop/lab/4 course/1 sem/AI/lab2/lr_2_task4.py:78: Mat
'labels' parameter of boxplot() has been renamed 'tick_labels' since Matplotl
e will be dropped in 3.11.
  plt.boxplot(results, labels=names)
2024-11-28 00:13:50.539 Python[14653:10324055] +[IMKClient subclass]: chose I
2024-11-28 00:13:50.539 Python[14653:10324055] +[IMKInputSession subclass]: c
```

2.5. Класифікація даних лінійним класифікатором Ridge.

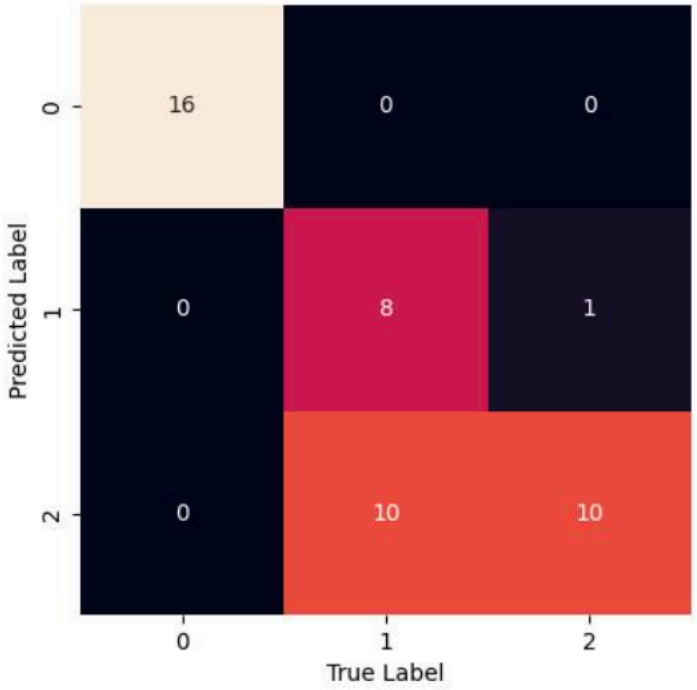
Лістинг програми:

```
import seaborn as sns
import numpy as np

from io import BytesIO
from matplotlib import pyplot as plt
```

```python
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import metrics

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)
ypred = clf.predict(X_test)

print('Accuracy:', np.round(metrics.accuracy_score(y_test, ypred), 4))
print('Precision:', np.round(metrics.precision_score(y_test, ypred,
average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(y_test, ypred,
average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(y_test, ypred,
average='weighted'), 4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(y_test,
ypred), 4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(y_test,
ypred), 4))
print('\t\tClassification Report:\n', metrics.classification_report(ypred,
y_test))

mat = confusion_matrix(y_test, 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")
```

Результат виконання:

```
(venv) venvantonosypchuk@MacBook-Air-Anton lab2 % python3 lr_2_task5.py
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
```



Налаштування класифікатора Ridge

tol=1e-2: Допуск для критерію зупинки. Це визначає точність оптимізації. Якщо зміна втраченого значення (loss) між ітераціями менша за це значення, оптимізація припиняється.

solver='sag': Алгоритм оптимізації "Stochastic Average Gradient Descent". Цей метод добре працює з великими наборами даних та ітеративно зменшує похибку, обробляючи міні-батчі даних.

RidgeClassifier є варіантом лінійного класифікатора, який додає регуляризацію L2 для боротьби з переобученням. Він підходить для класифікації задач, які вимагають балансування між простотою моделі та точністю.

Показники якості та результати

Accuracy: Частка правильно класифікованих прикладів.

Precision (зважена середня): Частка правильно передбачених позитивних класів серед усіх передбачень для кожного класу.

Recall (зважена середня): Частка правильно передбачених позитивних класів серед усіх фактичних позитивних зразків.

F1 Score (зважена середня): Гармонійне середнє Precision та Recall. Використовується для балансування між ними.

Cohen Kappa Score: Міра узгодженості між передбаченнями та реальними мітками, яка враховує випадкову згоду.

Matthews Corrcoef: Кореляційний коефіцієнт між передбаченнями та фактичними мітками. Він враховує дисбаланс у даних.

Пояснення зображення Confusion.jpg

Істинний клас 0 був правильно передбачений для 16 зразків. Істинний клас 1 був передбачений як 2 для 10 зразків, що вказує на значний confusion між цими класами. Матриця дозволяє зрозуміти, які класи модель плутає найчастіше.

Пояснення коефіцієнтів

Коефіцієнт Коена Каппа (Cohen Kappa Score) – це статистичний показник узгодженості між передбаченнями моделі та істинними мітками, з урахуванням випадкової згоди.

Коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient, MCC) – це міра якості класифікації, яка враховує всі чотири значення матриці плутанини (TP, TN, FP, FN). MCC підходить для задач із дисбалансом даних.