Міністерство освіти та науки України
Київський національний економічний університет
імені Вадима Гетьмана

Кафедра інформаційних систем
в економіці

Дисципліна **“Системи і методи штучного інтелекту”**

ЗВІТ

з лабораторної роботи №2

“Порівняння методів класифікації даних”

Підготував:

студент групи ІН-401

спеціальності “**Комп’ютерні науки**”

Михайлик В.А.

Київ - 2025

Хід роботи

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

Ознайомимося з набором даних.

```
#Імпортуємо бібліотеки
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

#Підключаємо вхідний файл, який містить дані
input_file = '/content/income_data.txt'

#Перетворюємо вхідний файл у датафрейм
df = pd.read_csv(input_file, header=None)

#Виведемо перші 10 рядків
df.head(10)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K

```
#Виводимо інформацію по даних у датафремі
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       32561 non-null  int64
 1   1       32561 non-null  object
 2   2       32561 non-null  int64
 3   3       32561 non-null  object
 4   4       32561 non-null  int64
 5   5       32561 non-null  object
 6   6       32561 non-null  object
 7   7       32561 non-null  object
 8   8       32561 non-null  object
 9   9       32561 non-null  object
 10  10      32561 non-null  int64
 11  11      32561 non-null  int64
 12  12      32561 non-null  int64
 13  13      32561 non-null  object
 14  14      32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

В результаті бачимо, що:

датафрейм складається з 15 колонок та з 32561 рядків, не має пропусків;

датафрем має 6 числових та 9 категоріальних колонок.

Продовжимо роботу щоб обчислити якість класифікації за показниками акуратності, повноти, точності та F1.

```
# Ініціалізуємо змінні для збереження даних та обмеження розміру вибірки
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

# Зчитуємо дані з текстового файлу пострічково, набираємо достатньо
прикладів кожного класу та розділяємо їх на 2 класи
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)

# Кодуємо всі текстові ознаки у числові значення
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])

# Розділяємо ознаки (X) та мітки (y)
X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

#Створюємо модель SVM-класифікатора
classifier = OneVsRestClassifier(LinearSVC(random_state=0))

#Навчання класифікатора
classifier.fit(X, y)

# Розбиваємо дані на навчальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Повторно створюємо та навчаємо класифікатор вже на тренувальних даних
classifier = OneVsRestClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, y_train)

# Передбачаємо результати на тестовій вибірці
y_test_pred = classifier.predict(X_test)

# Обчислення F-міри для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print('F1 score: ' + str(round(100*f1.mean(), 2)) + '%')
```

```
F1 score: 74.26%
```

```
# 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(item)
```

```
    else:
        input_data_encoded[i] = label_encoder[count].transform([item])[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])
```

```
<=50K
```

```
# Додаткові метрики для оцінки моделі на тестових даних
f1 = f1_score(y_test, y_test_pred, average='weighted')
precision = precision_score(y_test, y_test_pred, average='weighted')
accuracy = accuracy_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred, average='weighted')
print(f'F1: {f1*100:.2f}%')
print(f'Precision: {precision*100:.2f}%')
print(f'Accuracy: {accuracy*100:.2f}%')
print(f'Recall: {recall*100:.2f}%')
```

```
F1: 74.32%
Precision: 76.50%
Accuracy: 78.14%
Recall: 78.14%
```

Висновок: тестова точка належить до класу <=50K, тобто передбачуваний річний дохід становить менше або дорівнює 50 тисячам доларів. Це зумовлено рівнем освіти, типом роботи та іншими соціально-економічними ознаками.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

```
#Імпортуємо бібліотеки
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC, SVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

#Підключаємо вхідний файл, який містить дані
input_file = '/content/income_data.txt'

# Ініціалізуємо змінні для збереження даних та обмеження розміру вибірки
X = []
```

```
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

# Зчитуємо дані з текстового файлу пострічково, набираємо достатньо
прикладів кожного класу та розділяємо їх на 2 класи
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)

# Кодуємо всі текстові ознаки у числові значення
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])

# Розділяємо ознаки (X) та мітки (y)
X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

#Створення SVM-класифікаторів
classifiers = {
    'Linear SVM': LinearSVC(random_state=0),
    'RBF SVM': SVC(kernel='rbf', random_state=0),
    'Poly SVM (degree=3)': SVC(kernel='poly', degree=3, random_state=0),
    'Sigmoid SVM': SVC(kernel='sigmoid', random_state=0)
}

#Навчання класифікатора
for classifier in classifiers.items():
    print(f'\nClassifier: {classifier[0]}')
```

```
    clf = OneVsRestClassifier(classifier[1])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    clf.fit(X_train, y_train)
    y_test_pred = clf.predict(X_test)

# Додаткові метрики для оцінки моделі на тестових даних
    f1 = f1_score(y_test, y_test_pred, average='weighted')
    precision = precision_score(y_test, y_test_pred, average='weighted',
zero_division=0)
    accuracy = accuracy_score(y_test, y_test_pred)
    recall = recall_score(y_test, y_test_pred, average='weighted')
    print(f'F1: {f1*100:.2f}%')
    print(f'Precision: {precision*100:.2f}%')
    print(f'Accuracy: {accuracy*100:.2f}%')
    print(f'Recall: {recall*100:.2f}%')
```

```
Classifier: Linear SVM
F1: 74.32%
Precision: 76.88%
Accuracy: 78.34%
Recall: 78.34%

Classifier: RBF SVM
F1: 63.80%
Precision: 55.71%
Accuracy: 74.64%
Recall: 74.64%

Classifier: Poly SVM (degree=3)
F1: 63.80%
Precision: 55.71%
Accuracy: 74.64%
Recall: 74.64%

Classifier: Sigmoid SVM
F1: 63.68%
Precision: 63.56%
Accuracy: 63.82%
Recall: 63.82%
```

```
# 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(item)
    else:
        input_data_encoded[i] = label_encoder[count].transform([item])[0]
        count += 1

input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

# Для кожного класифікатора виконуємо навчання, передбачення класу для
тестової точки та вивід результату
for classifier in classifiers.values():
    print(f'\nClassifier: {classifier.__class__.__name__}')
    classifier.fit(X, y)
    predicted_class = classifier.predict(input_data_encoded)
```

```
    print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
Classifier: LinearSVC
 <=50K

Classifier: SVC
 <=50K

Classifier: SVC
 <=50K

Classifier: SVC
 <=50K
```

Висновок: за результатами тестування різних SVM моделей найкраще завдання класифікації виконує лінійна модель. Якщо обирати з нелінійних то найкраще працють класифікатори з поліноміальним та гаусовим ядром.

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

КРОК 1. ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ

```
# Імпортуємо набір даних Iris
from sklearn.datasets import load_iris
iris_dataset = load_iris()

# Виводимо ключі словника з даними
print('Ключі iris dataset: \n{}'.format(iris_dataset.keys()))
```

```
Ключі iris dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
# Виводимо частину опису датасету
print(iris_dataset['DESCR'][:193] + "\n...")
```

```
.. _iris_dataset:

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

**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive
...
```

```
# Виводимо назви класів (відповідей)
print('Назви відповідей: {}'.format(iris_dataset['target_names']))
```

```
Назви відповідей: ['setosa' 'versicolor' 'virginica']
```

```
# Виводимо назви ознак (характеристик)
print('Назви ознак: {}'.format(iris_dataset['feature_names']))
```

```
Назви ознак: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
# Виводимо тип об'єкта з ознаками
```

```
print('Тип масиву data: {}'.format(type(iris_dataset['data'])))
Тип масиву data: <class 'numpy.ndarray'>
```

```
# Виводимо розмірність масиву з ознаками
print('Форма масиву data: {}'.format(iris_dataset['data'].shape))
Форма масиву data: (150, 4)
```

```
# Виводимо перші 5 прикладів з даними
print('Перші п’ять рядків масиву
data:\n{}'.format(iris_dataset['data'][:5]))
Перші п’ять рядків масиву data:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

```
# Виводимо тип об'єкта з мітками класів
print('Тип масиву target: {}'.format(type(iris_dataset['target'])))
Тип масиву target: <class 'numpy.ndarray'>
```

```
# Виводимо всі мітки класів
print('Відповіді: {}'.format(iris_dataset['target']))
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 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 2 2 2 2
 2 2]
```

КРОК 2. ВІЗУАЛІЗАЦІЯ ДАНИХ

```
# Завантаження бібліотек

import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
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.multiclass import OneVsRestClassifier
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
```

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = pd.read_csv(url, names=names)

# shape
print(dataset.shape)
```

```
(150, 5)
```

```
# Зріз даних head
print(dataset.head(20))
```

	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
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa

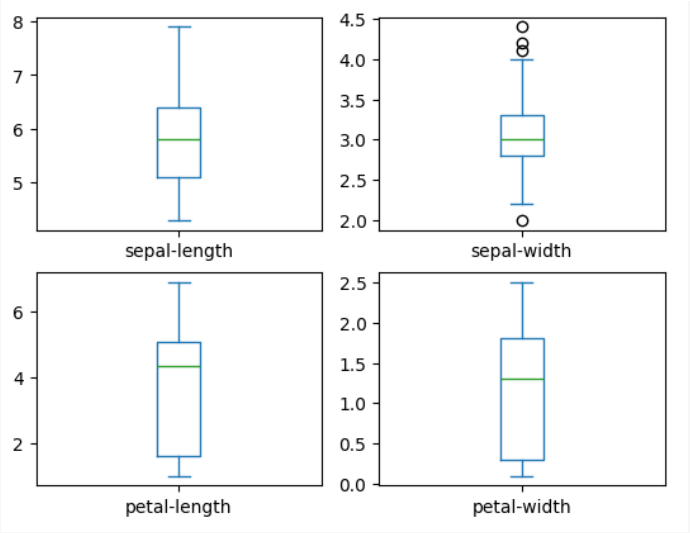
```
# Стастичні зведення методом describe
print(dataset.describe())
```

	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

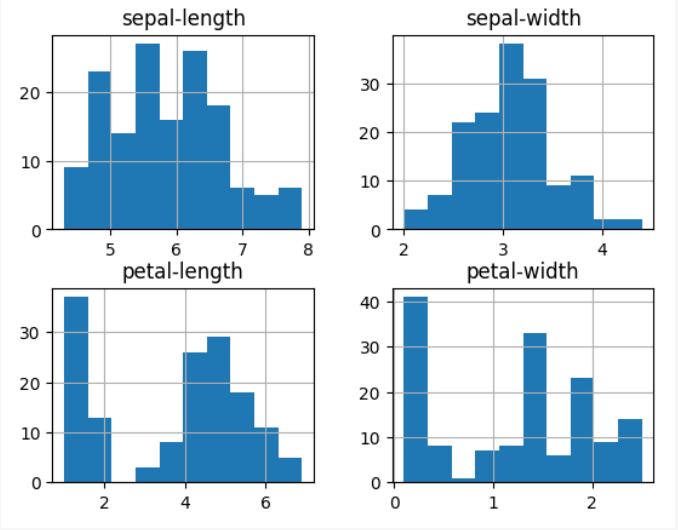
```
# Розподіл за атрибутом class
print(dataset.groupby('class').size())
```

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

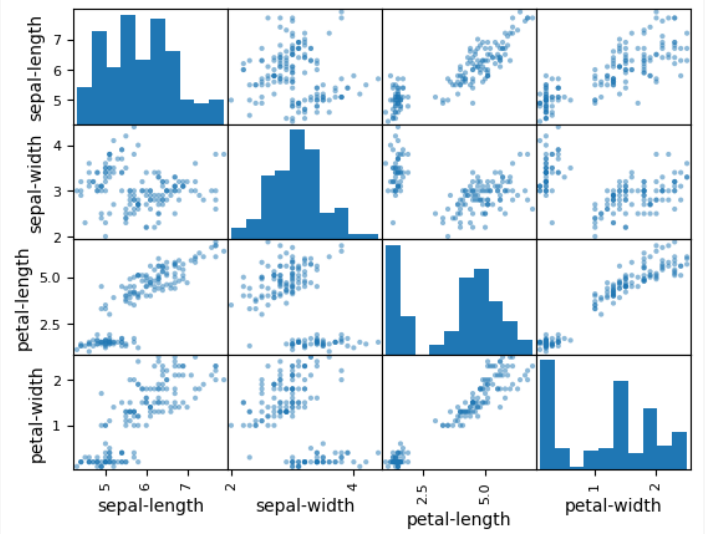
```
# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
plt.show()
```




```
# Гістограма розподілу атрибутів датасета
dataset.hist()
plt.show()
```




```
# Матриця діаграм розсіювання
scatter_matrix(dataset)
plt.show()
```



КРОК 3. СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ

```
#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]

# Розбиваємо дані на навчальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
```

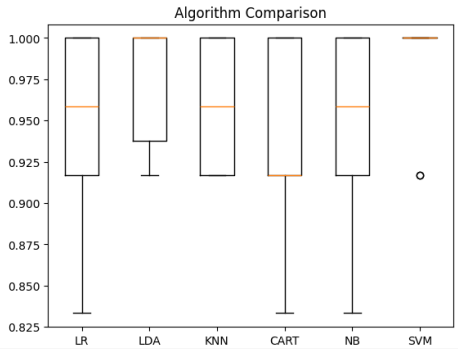
КРОК 4. КЛАСИФІКАЦІЯ (ПОБУДОВА МОДЕЛІ)

```
# Завантажуємо алгоритми моделі
models = []
models.append(('LR',
OneVsRestClassifier(LogisticRegression(solver='liblinear'))))
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, shuffle=True, random_state=1)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.053359)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

```
# Порівняння алгоритмів
plt.boxplot(results, labels=names)
plt.title('Algorithm Comparison')
plt.show()
```

```
<ipython-input-22-b599fd533b8d>:2: MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_labels' since Matplotlib 3.9; support for the old name will be dropped in 3.11.
  plt.boxplot(results, labels=names)
```



Висновок: найкращим методом класифікації є SVM, оскільки він показує найвищу медіану точності серед усіх алгоритмів. Також результати цього методу стабільні, без значних коливань чи викидів. Це свідчить про його надійність і високу ефективність на різних підмножинах даних.

КРОК 6. ОТРИМАННЯ ПРОГНОЗУ (ПЕРЕДБАЧЕННЯ НА ТРЕНУВАЛЬНОМУ НАБОРІ)

```
# Створюємо прогноз на контрольній вибірці
model = SVC(gamma='auto')
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

КРОК 7. ОЦІНКА ЯКОСТІ МОДЕЛІ

```
# Оцінюємо прогноз
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
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
```

Висновок: точність моделі на контрольній вибірці становить 96.67%, що є дуже високим результатом. Модель ідеально класифікує класи Iris-setosa і Iris-

virginica, а також майже безпомилково — Iris-versicolor. Загальні метрики (precision, recall, f1-score) на рівні 0.97 свідчать про стабільну та ефективну роботу класифікатора.

КРОК 8. ОТРИМАННЯ ПРОГНОЗУ (ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПЕРЕДБАЧЕННЯ)

```
# Створюємо нові дані для прогнозування (3 зразки, 4 ознаки)
X_new = np.array([[5, 2.9, 1, 0.2], [5.7, 4.2, 3, 1.7], [5.8, 2.9, 5, 1.9]])
print("форма масиву X_new: {}".format(X_new.shape))
```

```
форма масиву X_new: (3, 4)
```

```
# Виконуємо прогноз за допомогою навченої моделі
predictions = model.predict(X_new)
print("Прогноз: {}".format (predictions))
```

```
Прогноз: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

Висновок: у результаті отримання прогнозу з нових даних для 3 прикладів ми дізналися, що першому прикладу відповідає вид «Iris-setosa», другому – «Iris-versicolor», третьому – «Iris-virginica».

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

```
#Імпортуємо бібліотеки
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC, SVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from sklearn.multiclass import OneVsRestClassifier
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
```

```
from sklearn.model_selection import StratifiedKFold

#Підключаємо вхідний файл, який містить дані
input_file = '/content/income_data.txt'

# Ініціалізуємо змінні для збереження даних та обмеження розміру вибірки
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

# Зчитуємо дані з текстового файлу пострічково, набираємо достатньо
прикладів кожного класу та розділяємо їх на 2 класи
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(',')
        # print(data)

        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)

# Кодуємо всі текстові ознаки у числові значення
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)

#Створення SVM-класифікаторів
classifiers = {
    'Linear SVM': LinearSVC(random_state=0),
```

```
    'RBF SVM': SVC(kernel='rbf', random_state=0),
    'Poly SVM (degree=3)': SVC(kernel='poly', degree=3, random_state=0),
    'Sigmoid SVM': SVC(kernel='sigmoid', random_state=0)
}

# Навчання класифікатора
for classifier in classifiers.items():
    print(f'\nClassifier: {classifier[0]}')
    clf = OneVsRestClassifier(classifier[1])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    clf.fit(X_train, y_train)
    y_test_pred = clf.predict(X_test)

# Додаткові метрики для оцінки моделі на тестових даних
    f1 = f1_score(y_test, y_test_pred, average='weighted')
    precision = precision_score(y_test, y_test_pred, average='weighted',
zero_division=0)
    accuracy = accuracy_score(y_test, y_test_pred)
    recall = recall_score(y_test, y_test_pred, average='weighted')
    print(f'F1: {f1*100:.2f}%')
    print(f'Precision: {precision*100:.2f}%')
    print(f'Accuracy: {accuracy*100:.2f}%')
    print(f'Recall: {recall*100:.2f}%')
```

```
Classifier: Linear SVM
F1: 74.32%
Precision: 76.88%
Accuracy: 78.34%
Recall: 78.34%

Classifier: RBF SVM
F1: 63.80%
Precision: 55.71%
Accuracy: 74.64%
Recall: 74.64%

Classifier: Poly SVM (degree=3)
F1: 63.80%
Precision: 55.71%
Accuracy: 74.64%
Recall: 74.64%

Classifier: Sigmoid SVM
F1: 63.68%
Precision: 63.56%
Accuracy: 63.82%
Recall: 63.82%
```

```
# 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(item)
    else:
        input_data_encoded[i] = label_encoder[count].transform([item])[0]
        count += 1
```

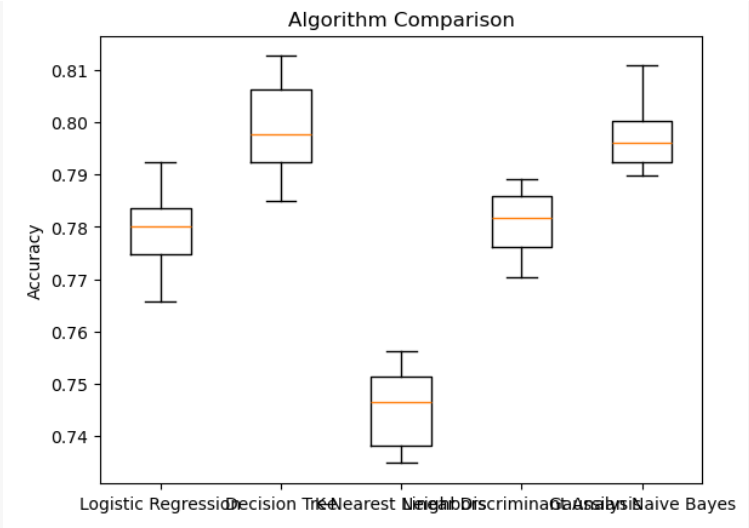
```
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

# Для кожного класифікатора виконуємо навчання, передбачення класу для
тестової точки та вивід результату
for classifier in classifiers.values():
    print(f'\nClassifier: {classifier.__class__.__name__}')
    classifier.fit(X, y)
    predicted_class = classifier.predict(input_data_encoded)
    print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
Classifier: LinearSVC
 <=50K

Classifier: SVC
 <=50K

Classifier: SVC
 <=50K

Classifier: SVC
 <=50K
```

```
# Завантажуємо алгоритми моделі
models = []
models.append(('LR',
OneVsRestClassifier(LogisticRegression(solver='liblinear'))))
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, shuffle=True, random_state=1)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.778938 (0.008510)
LDA: 0.780927 (0.006017)
KNN: 0.745410 (0.007315)
CART: 0.798831 (0.008858)
NB: 0.797132 (0.006025)
SVM: 0.752331 (0.000820)
```

```
# Порівняння алгоритмів
plt.boxplot(results, tick_labels=names)
plt.title('Algorithm Comparison')
plt.show()
```



Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

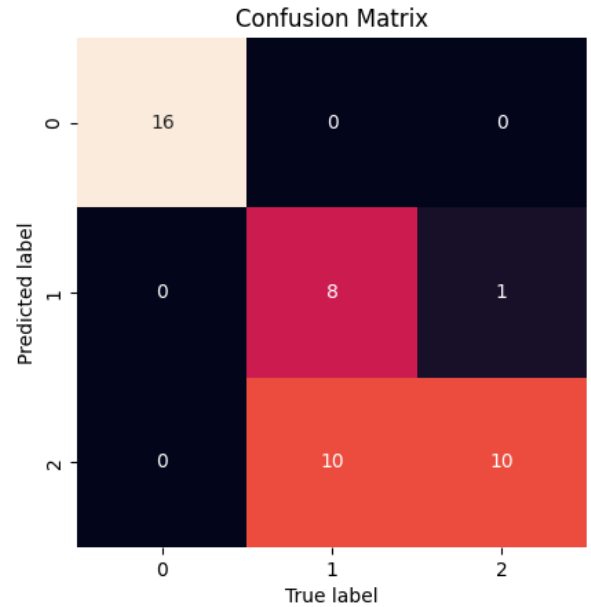
```
# Імпортуємо необхідні бібліотеки
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split

# Завантажуємо датасет Iris
iris = load_iris()

# Розділяємо дані на ознаки (X) та цільові значення (y)
X, y = iris.data, iris.target

# Ділимо вибірку на тренувальну і тестову (70%/30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

# Створюємо і навчаємо Ridge-класифікатор
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(X_train, y_train)

# Робимо прогноз на тестовій вибірці
y_pred = clf.predict(X_test)

# Імпортуємо метрики для оцінювання моделі
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, cohen_kappa_score, matthews_corrcoef, classification_report

# Виводимо значення основних метрик якості класифікації
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print("Cohen's Kappa:", cohen_kappa_score(y_test, y_pred))
```

```
print("Matthews Correlation Coefficient:", matthews_corrcoef(y_test,
y_pred))
print("\t\tClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7555555555555555
Precision: 0.8333333333333334
Recall: 0.7555555555555555
F1 Score: 0.7502986857825567
Cohen's Kappa: 0.6431146359048305
Matthews Correlation Coefficient: 0.6830664115990899
                Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.89      0.44      0.59        18
           2       0.50      0.91      0.65        11

    accuracy                           0.76        45
   macro avg       0.80      0.78      0.75        45
weighted avg       0.83      0.76      0.75        45
```

```
# Будуємо та візуалізуємо матрицю плутанини
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

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.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



Висновок: у цьому прикладі для класифікатора RidgeClassifier використано параметри tol=1e-2 та solver="sag". Параметр tol визначає поріг точності для зупинки ітерацій — чим менше значення, тим точніше, але довше триває навчання. Алгоритм sag (stochastic average gradient) є ефективним для великих наборів даних, оскільки працює швидше при великій кількості зразків.

Матриця плутанини показує як модель класифікувала зразки трьох класів. Модель правильно класифікувала всі зразки класу 0 (16), але плутає класи 1 і 2: 10 зразків класу 2 помилково віднесено до класу 1. Показники якості (accuracy, precision, recall, F1) допомагають оцінити точність моделі — наприклад, accuracy тут ~0.76, що свідчить про помірну ефективність класифікації.

Коефіцієнт Коена Каппа вимірює узгодженість між передбаченими та фактичними класами з урахуванням випадкових збігів — він показує, наскільки класифікація краща за випадкову. Коефіцієнт кореляції Метьюза (MCC) — це збалансований показник якості, який враховує всі значення матриці плутанини і добре підходить навіть при незбалансованих класах. У цьому випадку обидва коефіцієнти дають уявлення про загальну якість моделі: чим ближчі значення до 1, тим краще модель узгоджується з реальними мітками.

Посилання на GitHub:
https://github.com/VasylMykhailykUa/LR_2_Mykhailyk_Vasyl_IH-401