ЛАБОРАТОРНА РОБОТА № 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Виконала: Шевель Ольга ІПЗ-21-1, варіант 24

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати.

Хід роботи

Спочатку ознайомимося із набором даних та випишемо усі їх ознаки

Variable Name	Type	Demographic	Description	Missing Values
age	Числовий	Вік	N/A	no
workclass	Категоріальний	Тип зайнятості	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.	yes
fnlwgt	Числовий			no
education	Категоріальний	Освітній ступінь	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.	no
education-num	Числовий	Освітній ступінь		no
marital-status	Категоріальний	Сімейний стан	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.	no
occupation	Категоріальний	Спеціальність	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.	yes
relationship	Категоріальний	Статус у стосунках	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.	no
race	Категоріальний	Раса	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.	no
sex	Binary	Стать	Female, Male.	no

Variable Name	Type	Demographic	Description	Missing Values
capital-gain	Числовий			no
capital-loss	Числовий			no
hours-per-week	Числовий	Кількість робочих годин за тиждень		no
native-country	Категоріальний		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.	yes
income	Binary	Income	>50K, <=50K.	

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

```
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, cross_val_score
from sklearn.metrics import accuracy_score, recall_score, precision_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)
# y = np.array(y)

# Перетворення рядкових даних на числові
label_encoder = []
X_encoded = np.empty(X.shape)

for i, item in enumerate(X[0]):
    if item.dtype.kind in 'iufc':  # Numeric columns
        X_encoded[:, i] = X[:, i]
    else:
        le = preprocessing.LabelEncoder()
        label_encoder.append(le)
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))

# Навчання класифікатора
classifier.fit(X, y)

# Розділення на навчальний та тестовий набір
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))
classifier.fit(X_train, y_train)

# Прогнозування на тестовому наборі
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)) + "%")

# 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]]))
    count += 1

input_data_encoded = np.array([input_data_encoded])

# Використання класифікатора для кодованої точки даних
# та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(f"Класифікація кодованої точки: {label_encoder[-1].inverse_transform(predicted_class)[0]}\n");

# Обчислення accuracy_score
accuracy_s = accuracy_score(y_test, y_test_pred)
print(f"Accuracy: {accuracy_s * 100:.2f}%")

# Обчислення повноти
recall = recall_score(y_test, y_test_pred)
print(f"Recall RF: {recall * 100:.2f}%")
# Обчислення точності
precision = precision_score(y_test, y_test_pred)
print(f"Precision RF: {precision * 100:.2f}%")
```

```
C:\Users\OlhaShevel\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\O
F1 score: 74.27%
Класифікація кодованої точки: <=50K

Accuracy: 78.14%
Recall RF: 26.54%
Precision RF: 68.62%
```

Тестова точка належить до другого класу

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

1. Поліноміальне ядро

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

не дочекалася результатів, багато часу

2. Гаусове ядро

```
classifier = svm.SVC(kernel='rbf', gamma=0.1, C=10.0)
```

3. Сигмоїдальне ядро

```
classifier = svm.SVC(kernel='sigmoid')
```

```
C:\Users\OlhaShevel\AppData\Local\Programs\Python\Python311\python.exe "C:\Use
F1 score: 62.98%
Класифікація кодованої точки: >50K

Accuracy: 60.82%
Recall RF: 24.21%
Precision RF: 23.77%

Process finished with exit code 0
```

Висновки: Найкраще ядро для класифікації залежить від характеру даних. Якщо дані є лінійно роздільними, лінійне ядро буде найкращим вибором. Для більш складних і нелінійних даних, гаусове ядро (RBF) зазвичай показує найкращі результати.

На даному прикладі вдалося побвчити лише результати лінійного та сигмоїдального ядер. І лінійне показало вищу точність та кращі результати. Інші методи можливо призначені для даних іншого виду, тому їх використання займає значні часові ресурси.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

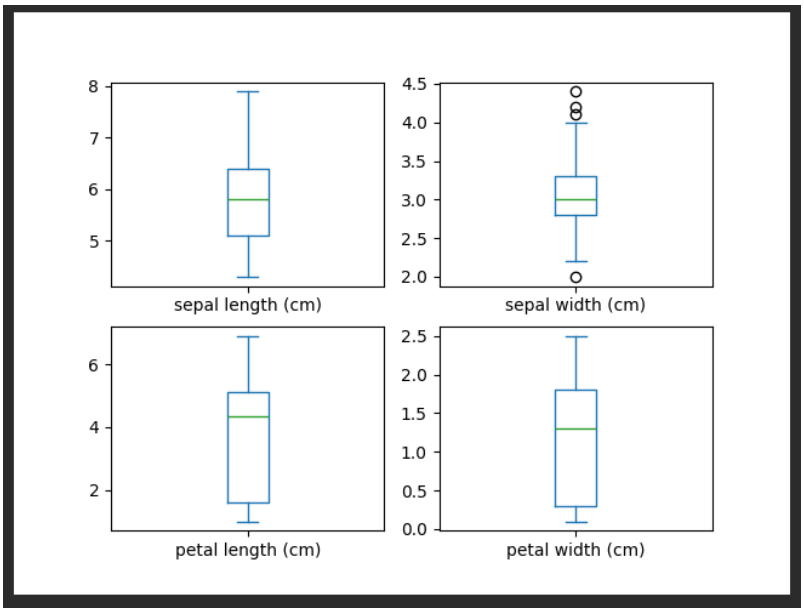
КРОК 1. ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ

```
from sklearn.datasets import load_iris
import pandas as pd

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))

iris_df = pd.DataFrame(data=iris_dataset.data, columns=iris_dataset.feature_names)

# Виведення значень ознак для перших п'яти прикладів
print(iris_df.head())

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

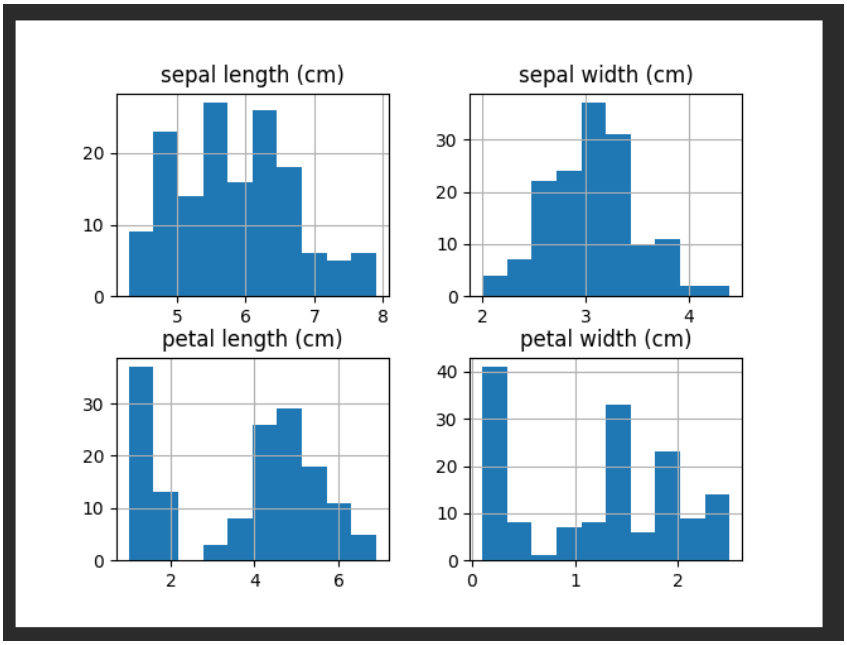
```
C:\Users\OlhaShevel\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\OlhaShevel\OneDrive - Spher
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

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)
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
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 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

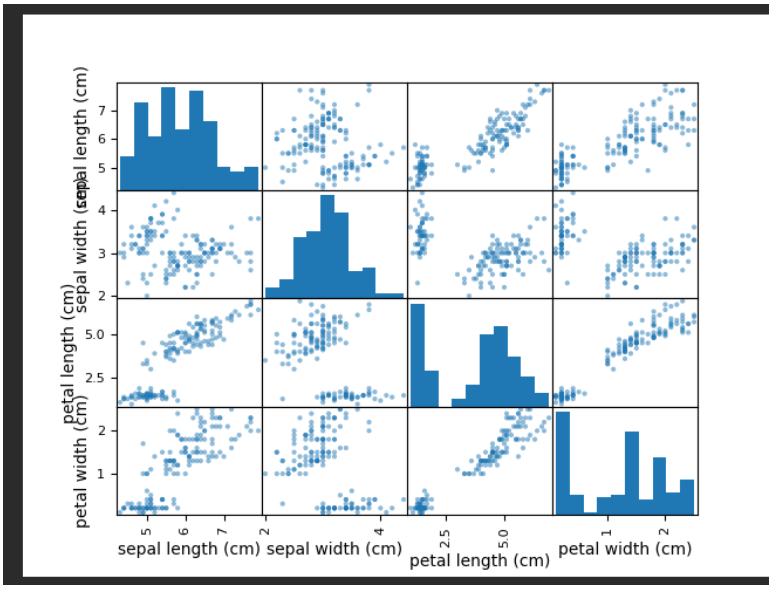
```
# Діаграма розмаху
iris_df.plot(kind='box', subplots=True, layout=(2, 2),
             sharex=False, sharey=False)
pyplot.show()
```




```
# Гістограма розподілу атрибутів датасета
iris_df.hist()
pyplot.show()
```




```
#Матриця діаграм розсіювання
pd.plotting.scatter_matrix(iris_df)
pyplot.show()
```



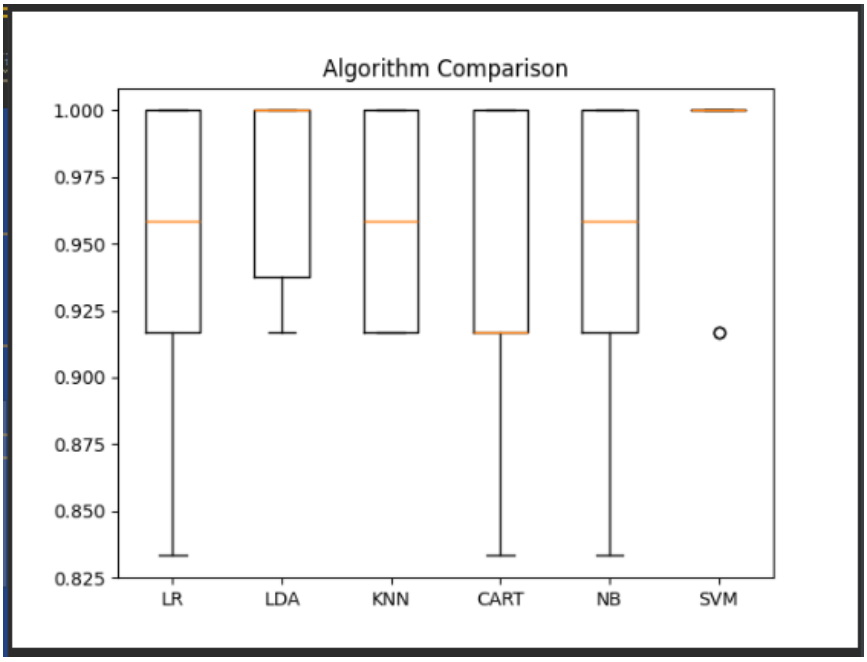
КРОК 3. СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ

```
# Розділення датасету на навчальну та контрольну вибірки
array = iris_df.values
# Вибір перших 4-х стовпців
X = array[:, 0:4]
# Вибір 5-го стовпця
Y = array[:, 4]
```

```
# Розділення X и y на навчальну и контрольну вибірки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=0.20, random_state=1)
```

КРОК 4. КЛАСИФІКАЦІЯ (ПОБУДОВА МОДЕЛІ)

```
# 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()
```



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_validation)
```

КРОК 7. ОЦІНКА ЯКОСТІ МОДЕЛІ

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

КРОК 8. ОТРИМАННЯ ПРОГНОЗУ (ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПЕРЕДБАЧЕННЯ)

```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("форма масиву X_new: {}".format(X_new.shape))

# prediction = KNeighborsClassifier().fit(X_train, Y_train).predict(X_new) # код із моделлю із прикладу KNN
prediction = SVC(gamma='auto').fit(X_train, Y_train).predict(X_new) # код із моделлю SVC

print("Прогноз: {}".format(prediction))

predicted_class = iris_dataset['target_names'][int(prediction[0])]
print("Спрогнозований клас: {}".format(predicted_class))
```

```
форма масиву X_new: (1, 4)
Прогноз: [0.]
Спрогнозований клас: setosa
```

Висновок: **SVM** є найкращим методом у цьому порівнянні, оскільки має найвищу середню точність (0.9833) і найменше стандартне відхилення (0.0333), що вказує на стабільність і ефективність цього методу. Квітка належить до класу setosa.

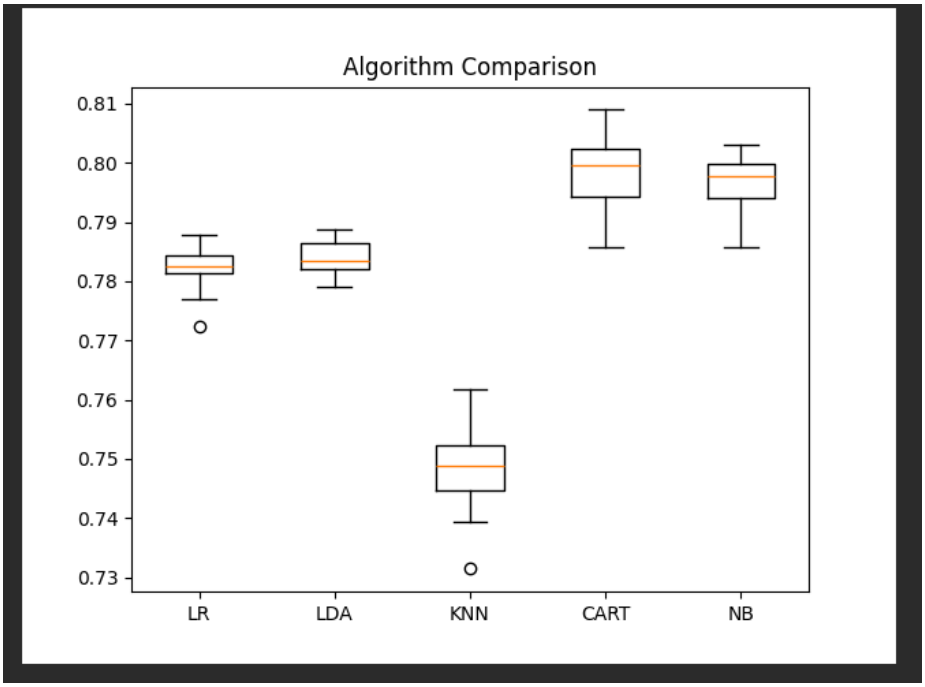
Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

```
from sklearn.model_selection import train_test_split
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 cross_val_score
from sklearn.model_selection import StratifiedKFold
from matplotlib import pyplot
import numpy as np
```

```
from sklearn import preprocessing

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.dtype.kind in 'iufc':  # Numeric columns
        X_encoded[:, i] = X[:, i]
    else:
        le = preprocessing.LabelEncoder()
        label_encoder.append(le)
        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=0.20, 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()))

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

```
LR: 0.781964 (0.004251)
LDA: 0.784077 (0.003008)
KNN: 0.748228 (0.008123)
CART: 0.799577 (0.005253)
NB: 0.796593 (0.004893)
```



Час роботи моделі SVM виявився завеликим, тому цю модель не було включено до порівняння

Як можна побачити, найвищу середню точність має алгоритм CART, що дозволяє вважати його найркащим серед інших для цього набору даних.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Виправлений код

```
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
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)
from sklearn import metrics

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")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format="svg")
```

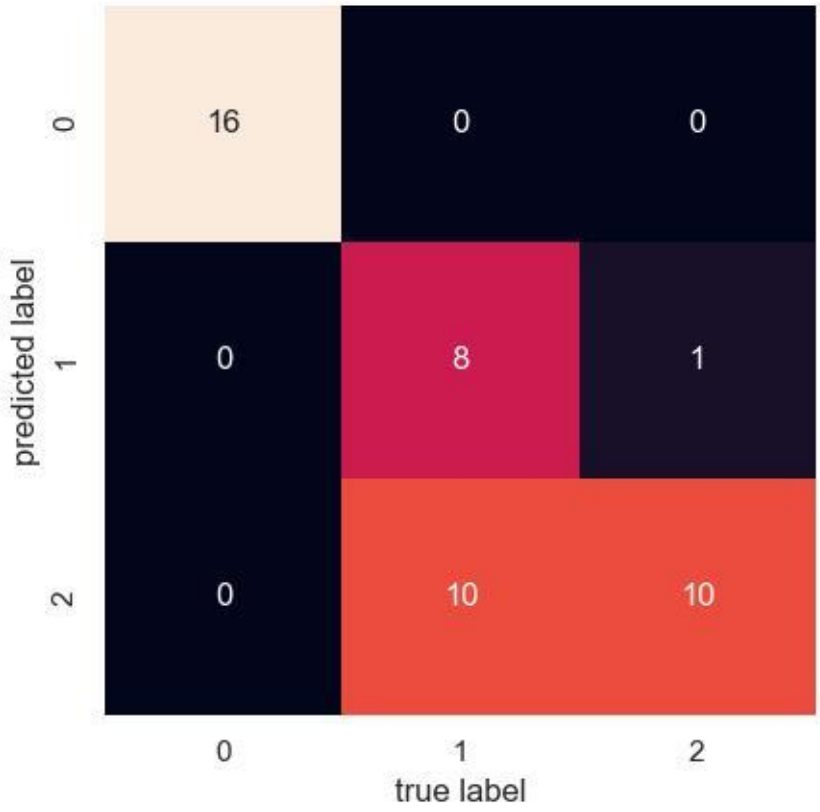
В цьому завданні використовується класифікатор **RidgeClassifier**. Цей класифікатор спочатку перетворює цільові значення в {-1, 1}, а потім розглядає проблему як завдання регресії (регресія з кількома виходами у випадку з кількома класами). Використані наступні налаштування:

1) **tol=1e-2** - параметр для визначення критерію зупинки алгоритму оптимізації. Якщо зміна втрат стає меншою за цей поріг, алгоритм зупиняється. Значення 1e-2 означає, що зупинка відбудеться, коли зміна втрат стане менше 0.01.
2) **solver="sag"** - параметр, що визначає метод оптимізації. Використовується метод **"sag"** (Stochastic Average Gradient Descent),

який є варіантом стохастичного градієнтного спуску, що ефективно працює з великими наборами даних.

Показник	Значення	Пояснення
Accuracy	0.75 -> 75%	Оцінює частку правильних передбачень серед усіх передбачень.
Precision	0.83 -> 83%	Це частка правильних передбачень позитивних класів від усіх передбачених позитивних прикладів. В даному випадку використовується зважена середня для усіх класів.
Recall	0.75 -> 75%	Частка правильних позитивних передбачень серед усіх передбачених позитивних прикладів.
F1 score	0.75 -> 75%	Гармонічне середнє між Precision і Recall
Cohen Kappa Score	0.64 -> 64%	Оцінює рівень згоди між передбаченнями та реальними значеннями, враховуючи випадкову згоду.
Matthews Corrcoef	0.68 -> 68%	Враховує всі елементи матриці плутанини, показує якість класифікації, навіть при класовому дисбалансі.

Зображення Confision.jpg



Це зображення є відображенням **матрицею плутанини (confusion matrix)**, яка показує результат роботи моделі класифікації. Кожен елемент матриці відповідає кількості передбачень для кожної пари реальних та передбачених класів.

Інтерпетація цієї матриці наступна:

- **Рядки** – це передбачені класи
- **Стовпці** - це реальні класи (мітки істиних значень)
- На діагоналі знаходяться кількості правильно розрахованих моделлю значень кожного класу.

Є певні помилки між класами 1 і 2. Модель часто плутає ці класи, що може свідчити про подібність між їх характеристиками в даному наборі даних.

Коефіцієнт Коена Каппа показує, наскільки результат моделі перевершує класифікацію випадковим чином. На цьому прикладі значення 64% вказує на не досить високу згоду між передбаченнями моделі та фактичними результатами, тобто модель доволі часто вгадує правильні значення.

MCC (коефіцієнт кореляції Метью) — це статистичний показник, який зазвичай використовується для двійкової класифікації, коливається від -1 до 1 і розширює його відповідність на всі значення матриці плутанини, тобто. Він повертає хороші результати, лише якщо результати всіх чотирьох значень матриці плутанини (tp, tn, fp, fn) є хорошими всупереч точності та оцінці f1. По суті, ми можемо сказати, що це показник найбільш надійної та правдивої оцінки. Значення **0.68** вказує на те, що модель досить добре класифікує, але є ще простір для покращення . Цей результат перевищує випадкову класифікацію, але модель все ж таки робить певну кількість помилок, як видно з матриці плутанини.

Посилання на Github репозиторій:
https://github.com/missShevel/SHI_Shevel_Olha_IPZ-21-1/tree/master/Lab2