Лабораторна робота №2

Тема: ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

Хід роботи:

Завдання 2.1. Класифікація за допомогою машин опорних векторів (SVM). Ознайомитись з набором даних.

№	Назва ознаки	Що позначає	Вид (числові чи категоріальні)
1	age	Вік	Цілочисельна
2	workclass	Тип зайнятості / клас працівника	Категоріальна
3	fnlwgt	“Final weight” — ваговий коефіцієнт запису	Цілочисельна
4	education	Рівень освіти	Категоріальна
5	education-num	Кількість років освіти	Цілочисельна
6	marital-status	Сімейний/цивільний статус	Категоріальна
7	occupation	Сфера чи професія / тип роботи	Категоріальна
8	relationship	Тип стосунку/родинного стану в сім’ї	Категоріальна
9	race	Раса особи	Категоріальна
10	sex	Стать	Бінарна
11	capital-gain	Дохід від капіталу / приріст капіталу	Цілочисельна

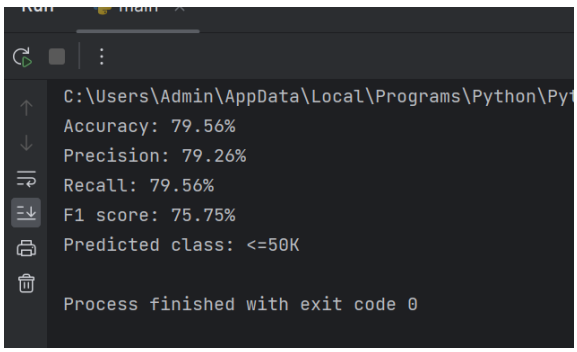
12	capital-loss	Збиток від капіталу / втрати капіталу	Цілочисельна
13	hours-per-week	Кількість годин роботи на тиждень	Цілочисельна
14	native-country	Рідна країна / країна походження особи	Категоріальна

Лістинг програми:

```
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
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

input_file = 'income_data.txt'

X = []
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.strip().split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

X = np.array(X)

label_encoder = []
X_encoded = np.empty(X.shape, dtype=int)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1]
y = X_encoded[:, -1]

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, max_iter=10000))

classifier.fit(X_train, y_train)

y_test_pred = classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted')
recall = recall_score(y_test, y_test_pred, average='weighted')
f1 = f1_score(y_test, y_test_pred, average='weighted')


print("Accuracy: {:.2f}%".format(100 * accuracy))
print("Precision: {:.2f}%".format(100 * precision))
print("Recall: {:.2f}%".format(100 * recall))
print("F1 score: {:.2f}%".format(100 * f1))

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(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)


predicted_class = classifier.predict(input_data_encoded)
print("Predicted class:", label_encoder[-
1].inverse_transform(predicted_class)[0])
```

Результат:



Висновок: тестова точка з вхідними даними ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States'] належить до класу "<=50K"**.** Це означає, що класифікатор передбачив для цієї людини дохід менше або рівно 50 тис. доларів на рік**.**

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

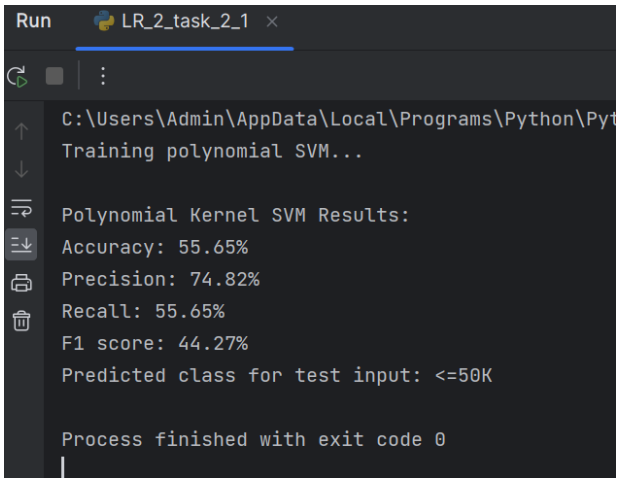
З поліноміальним ядром:

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

input_file = 'income_data.txt'


X = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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.strip().split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

X = np.array(X)

label_encoder = []
X_encoded = np.empty(X.shape, dtype=int)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1]
y = X_encoded[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Поліноміальне ядро
classifier = SVC(kernel='poly', degree=3, gamma='scale')  # degree можна змінити

print("Training polynomial SVM...")
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("\nPolynomial Kernel SVM Results:")
print("Accuracy: {:.2f}%".format(100 * accuracy))
print("Precision: {:.2f}%".format(100 * precision))
print("Recall: {:.2f}%".format(100 * recall))
print("F1 score: {:.2f}%".format(100 * f1))

# Тестова точка
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
              'Handlers-cleaners', 'Not-in-family', 'White', 'Male',
              '0', '0', '40', 'United-States']

input_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_encoded[i] = int(item)
    else:
        input_encoded[i] = label_encoder[count].transform([item])[0]
        count += 1

input_encoded = np.array(input_encoded).reshape(1, -1)
predicted_class = classifier.predict(input_encoded)
print("Predicted class for test input:", label_encoder[-
1].inverse_transform(predicted_class)[0])
```

Результат:



З Гаусовим ядром:

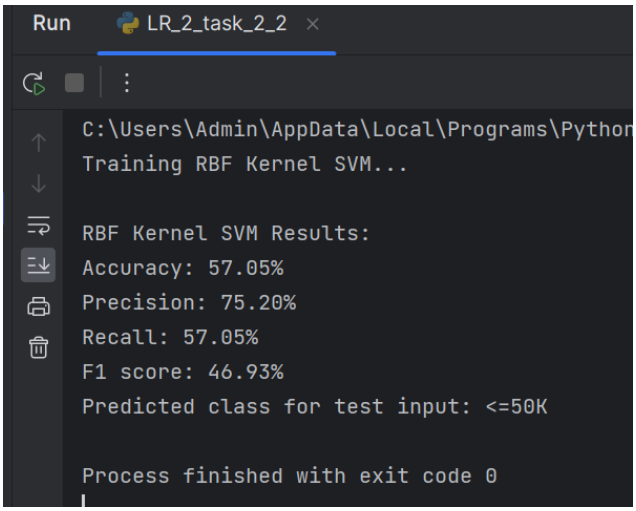
Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

input_file = 'income_data.txt'
```

```
X = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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.strip().split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

X = np.array(X)

label_encoder = []
X_encoded = np.empty(X.shape, dtype=int)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1]
y = X_encoded[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

classifier = SVC(kernel='rbf', gamma='scale')

print("Training RBF Kernel SVM...")
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("\nRBF Kernel SVM Results:")
print("Accuracy: {:.2f}%".format(100 * accuracy))
print("Precision: {:.2f}%".format(100 * precision))
print("Recall: {:.2f}%".format(100 * recall))
print("F1 score: {:.2f}%".format(100 * f1))

# Тестова точка
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
              'Handlers-cleaners', 'Not-in-family', 'White', 'Male',
              '0', '0', '40', 'United-States']
```

```
input_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_encoded[i] = int(item)
    else:
        input_encoded[i] = label_encoder[count].transform([item])[0]
        count += 1

input_encoded = np.array(input_encoded).reshape(1, -1)
predicted_class = classifier.predict(input_encoded)
print("Predicted class for test input:", label_encoder[-
1].inverse_transform(predicted_class)[0])
```

Результат:



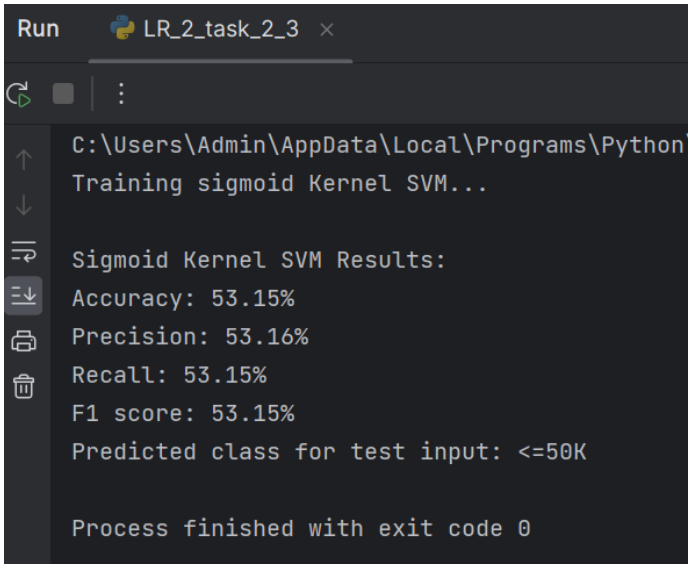
З сигмоїдальним ядром:

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score


input_file = 'income_data.txt'

X = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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.strip().split(', ')
```

```
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

X = np.array(X)

label_encoder = []
X_encoded = np.empty(X.shape, dtype=int)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1]
y = X_encoded[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

classifier = SVC(kernel='sigmoid', gamma='scale')

print("Training sigmoid Kernel SVM...")
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("\nSigmoid Kernel SVM Results:")
print("Accuracy: {:.2f}%".format(100 * accuracy))
print("Precision: {:.2f}%".format(100 * precision))
print("Recall: {:.2f}%".format(100 * recall))
print("F1 score: {:.2f}%".format(100 * f1))

# Тестова точка
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
              'Handlers-cleaners', 'Not-in-family', 'White', 'Male',
              '0', '0', '40', 'United-States']

input_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_encoded[i] = int(item)
    else:
        input_encoded[i] = label_encoder[count].transform([item])[0]
        count += 1

input_encoded = np.array(input_encoded).reshape(1, -1)
predicted_class = classifier.predict(input_encoded)
print("Predicted class for test input:", label_encoder[-
1].inverse_transform(predicted_class)[0])
```

Результат:



Висновок: серед протестованих SVM з різними ядрами найкращі результати для даного набору даних показало RBF (Гаусове) ядро, яке забезпечує найвищу точність (точність класифікації) та найкращий баланс між загальною правильністю (достовірністю класифікації) і повнотою. Поліноміальне ядро продемонструвало трохи гірші показники, а сигмоїдальне ядро показало найнижчу ефективність і не підходить для цього завдання. Незалежно від вибору ядра, тестова точка була класифікована як <=50K.

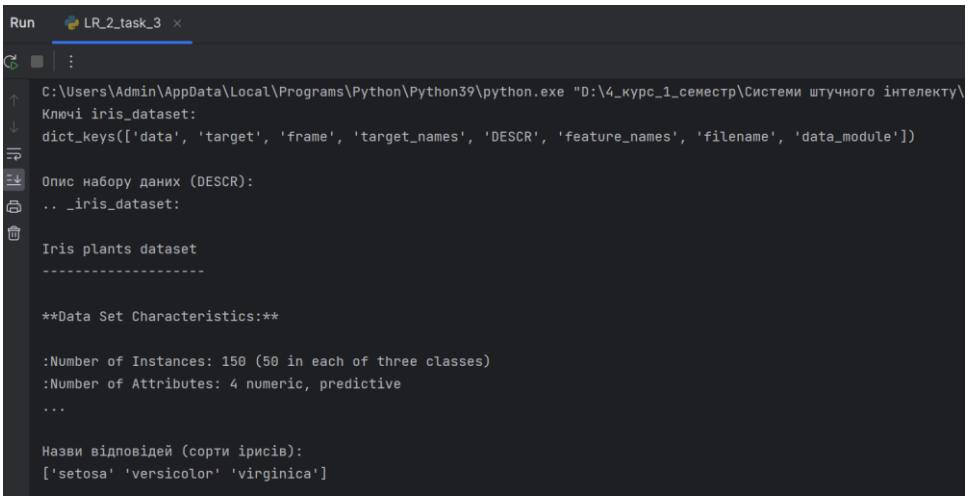
Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

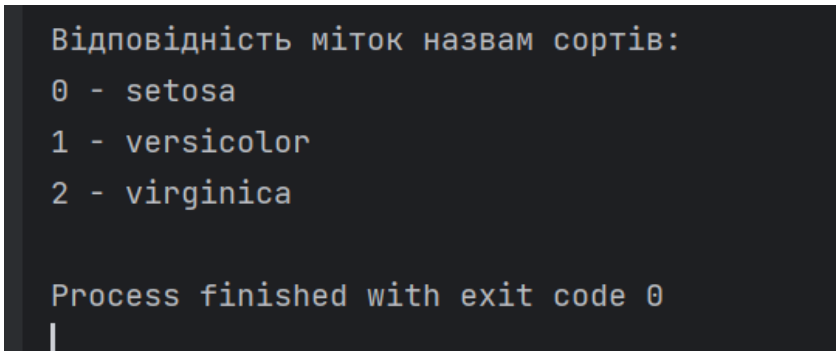
Лістинг програми:

```
from sklearn.datasets import load_iris
iris_dataset = load_iris()


print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))


print("\nОпис набору даних (DESCR):")
print(iris_dataset['DESCR'][:193] + "\n...")


print("\nНазви відповідей (сорти ірисів):")
print(iris_dataset['target_names'])


print("\nНазва ознак: ")
print(iris_dataset['feature_names'])
```

```
print("\nТип масиву data:", type(iris_dataset['data']))
print("Форма масиву data:", iris_dataset['data'].shape)


print("\nПерші 5 прикладів даних (X):")
print(iris_dataset['data'][:5])

print("\nТип масиву target:", type(iris_dataset['target']))
print("Відповіді (мітки) для всіх квіток:")
print(iris_dataset['target'][:5], "...")
print("Відповідність міток назвам сортів:")
for i, name in enumerate(iris_dataset['target_names']):
    print(f"{i} - {name}")
```

Результат:





Крок 2:

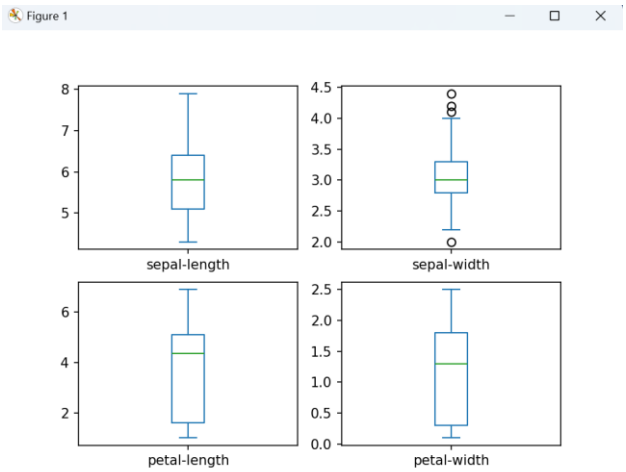
```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
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


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("Форма датасету (рядки, стовпці):")
print(dataset.shape)

print("\nПерші 20 рядків:")
print(dataset.head(20))

print("\nСтатистичні зведення:")
print(dataset.describe())

print("\nКількість екземплярів у кожному класі:")
print(dataset.groupby('class').size())

# Діаграма розмаху (BoxPlot)
print("\nПобудова діаграми розмаху...")
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False,
sharey=False)
pyplot.show()

# Гістограми
print("\nГістограми...")
dataset.hist()
pyplot.show()

# Діаграма розсіювання (scatter matrix)
print("\nМатриця діаграм розсіювання...")
scatter_matrix(dataset)
pyplot.show()
```

Результат:

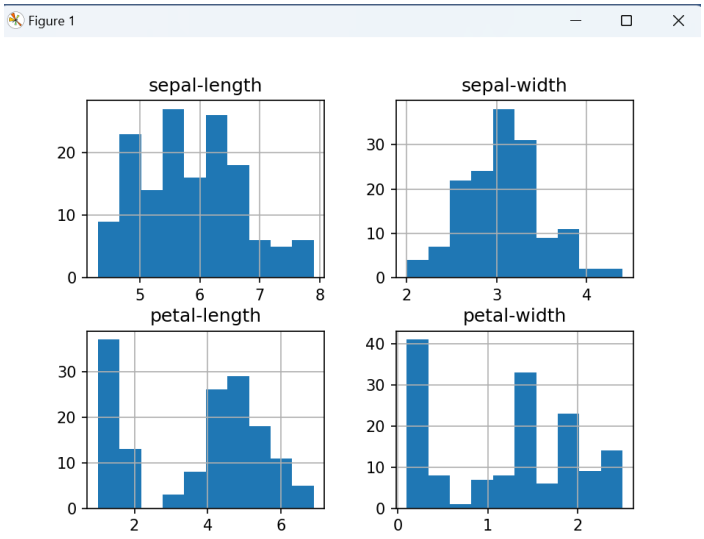
```
Run  LR_2_task_3_new
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe "D:\4_курс_1_семестр\C
Форма датасету (рядки, стовпці):
(150, 5)

Перші 20 рядків:
    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
```

```
Run  LR_2_task_3_new
Статистичні зведення:
       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

Кількість екземплярів у кожному класі:
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64

Побудова діаграми розмаху...
```

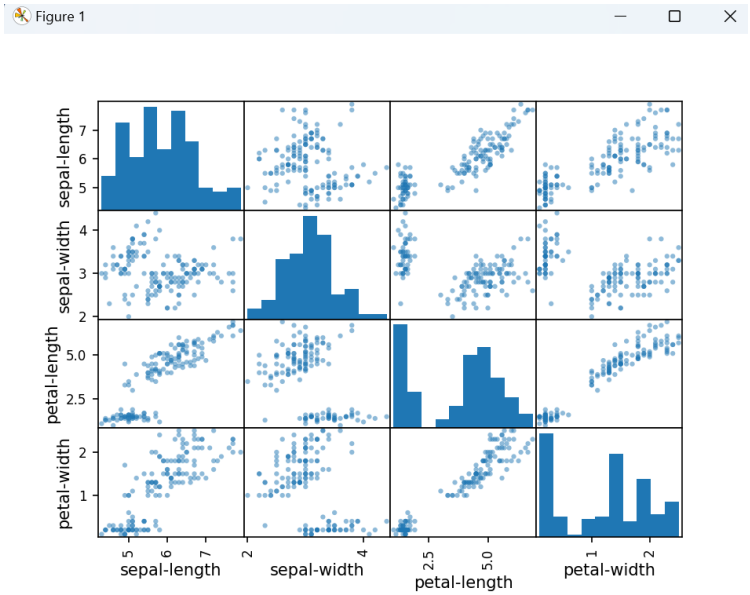
Діаграма розмаху (BoxPlot):



Гістограми:



Діаграма розсіювання (scatter matrix):



Крок 3-4: СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ. КЛАСИФІКАЦІЯ (ПОБУДОВА МОДЕЛІ).

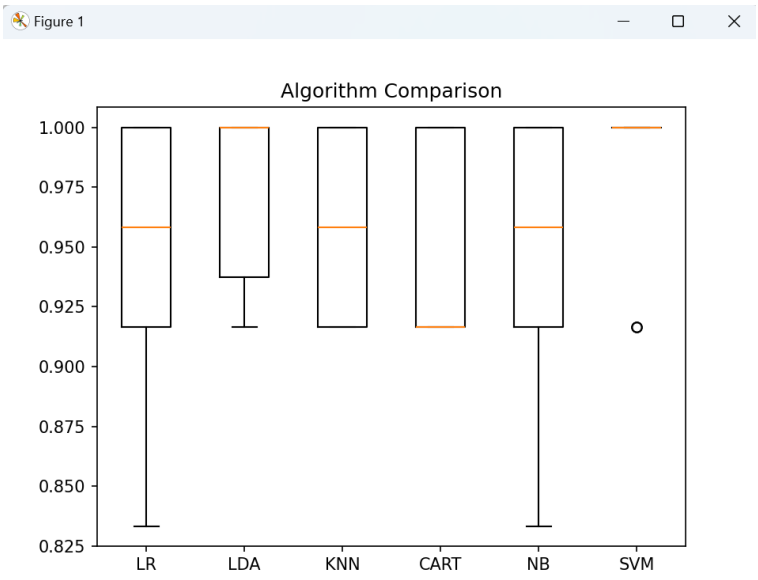
Лістинг програми:

```
from pandas import read_csv
from matplotlib import pyplot as plt
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
```

```
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)

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, shuffle=True
)


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()))

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

best_model = LogisticRegression(solver='liblinear', multi_class='ovr')
best_model.fit(X_train, Y_train)
predictions = best_model.predict(X_validation)

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

print("\nAccuracy on test set:", accuracy_score(Y_validation, predictions))
print("\nConfusion Matrix:\n", confusion_matrix(Y_validation, predictions))
print("\nClassification Report:\n", classification_report(Y_validation,
predictions))
```

Результат:

```
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)
```



Я вважаю найкраща модель за цими даними - SVM з точністю 98,33% і найменшим стандартним відхиленням 0,0333, тому вона є найбільш стабільною та точною для класифікації ірисів.

Крок 5 - 8: ОПТИМІЗАЦІЯ ПАРАМЕТРІВ МОДЕЛІ. ОТРИМАННЯ ПРОГНОЗУ (ПЕРЕДБАЧЕННЯ НА ТРЕНУВАЛЬНОМУ НАБОРІ). ОЦІНКА ЯКОСТІ МОДЕЛІ. ОТРИМАННЯ ПРОГНОЗУ (ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПЕРЕДБАЧЕННЯ).

Лістинг програми:

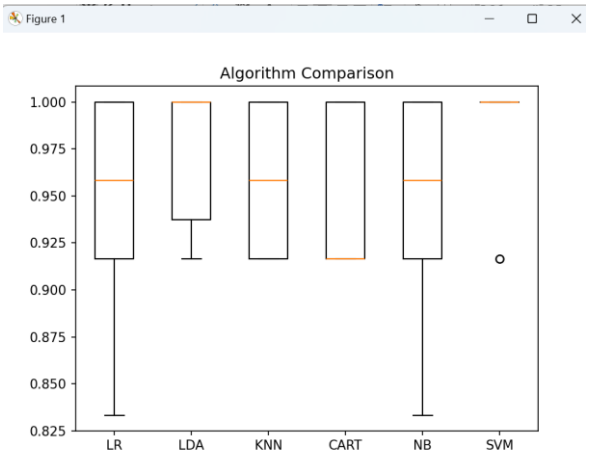
```
import numpy as np
from pandas import read_csv
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split, StratifiedKFold,
cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.linear_model import LogisticRegression
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


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)

# ----------------------------------------------------------
# КРОК 3. РОЗДІЛЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРУ
# ----------------------------------------------------------
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, shuffle=True)

# ----------------------------------------------------------
# КРОК 4. КЛАСИФІКАЦІЯ І ПОРІВНЯННЯ АЛГОРИТМІВ
# ----------------------------------------------------------
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()))

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

# ----------------------------------------------------------
# КРОК 6. ПЕРЕДБАЧЕННЯ НА КОНТРОЛЬНІЙ ВИБІРЦІ
# ----------------------------------------------------------
best_model = SVC(gamma='auto')
best_model.fit(X_train, Y_train)
predictions = best_model.predict(X_validation)

# ----------------------------------------------------------
# КРОК 7. ОЦІНКА ЯКОСТІ МОДЕЛІ
# ----------------------------------------------------------
print("Accuracy на тестовому наборі:", accuracy_score(Y_validation,
predictions))
print("Confusion Matrix:")
print(confusion_matrix(Y_validation, predictions))
print("Classification Report:")
print(classification_report(Y_validation, predictions))

# ----------------------------------------------------------
# КРОК 8. ПЕРЕДБАЧЕННЯ ДЛЯ НОВОЇ КВІТКИ
# ----------------------------------------------------------
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масиву X_new:", X_new.shape)

prediction_new = best_model.predict(X_new)
print("Прогнозоване значення класу:", prediction_new)
```

Результат:

```
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)
Accuracy на тестовому наборі: 0.9666666666666667
Confusion Matrix:
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
```

```
Classification Report:
                 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 X_new: (1, 4)
Прогнозоване значення класу: ['Iris-setosa']
```



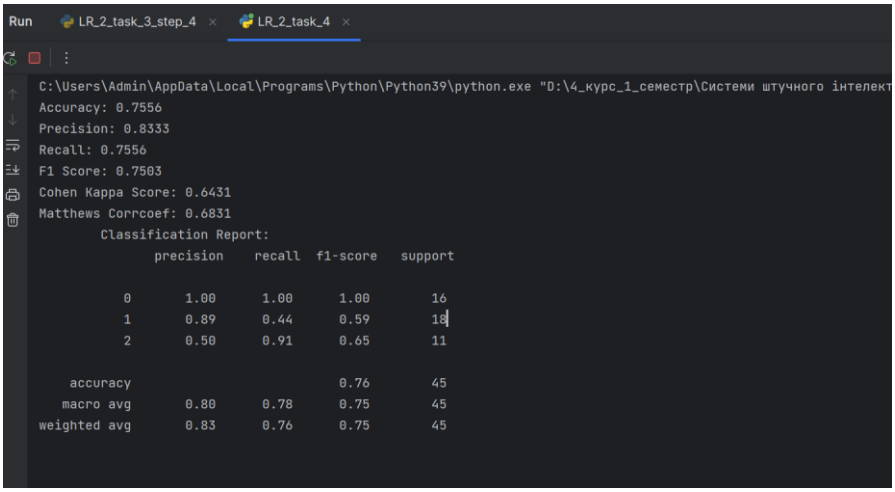
Під час тренування моделей на навчальному наборі найвищу точність показала модель SVM - 98,3%. На тестовому наборі точність класифікації склала 96,7%, що свідчить про хорошу узагальнювальну здатність моделі. Квітка з параметрами [5, 2.9, 1, 0.2] була віднесена моделлю до класу Iris-setosa.

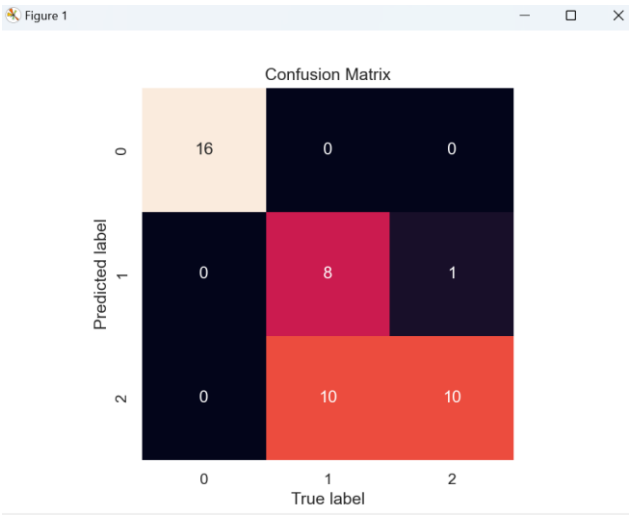
Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from io import BytesIO

iris = load_iris()
X, y = iris.data, iris.target

Xtrain, Xtest, 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(Xtest)

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(ytest,
ypred))

mat = confusion_matrix(ytest, ypred)
sns.set()
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('True label')
plt.ylabel('Predicted label')
plt.title('Confusion Matrix')
plt.savefig("Confusion.jpg")
plt.show()

f = BytesIO()
plt.savefig(f, format="svg")
```

Результат:





Дане зображення це - матриця плутанини (Confusion Matrix).

Це таблиця, яка показує, як добре штучний інтелект вгадує правильні відповіді, коли йому дають тестові дані.

Рядки ("Predicted label") - це те, що вгадав.

Стовпці ("True label") - це те, яким був правильний сорт насправді.

Числа по діагоналі (16, 8, 10): це правильні відповіді. Наприклад, 16 разів класифікатор сказав «Це Сорт 0» і він справді був сортом 0.

Числа поза діагоналлю (10 і 1): це помилки.

В цілому класифікатор працює добре (точність 75.56%), але всеодно має проблеми:

- Сорт 0: вгаданий ідеально (16 з 16).
- Сорти 1 і 2: тут виникла плутанина.

Коли насправді був сорт 1 (всього їх 18), класифікатор вгадав лише 8 з них.

10 разів він помилково сказав, що це сорт 2.

Тобто, коли ШІ бачить сорт 1, він часто плутає його з сортом 2. Це підтверджує, що для сорту 1 дуже низький показник повноти (Recall = 0.44).

Висновок: під час виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python ми дослідили різні методи класифікації даних та навчилися їх порівнювати.

Посилання на git: https://github.com/KokhanTetiana/AI-Systems