ЛАБОРАТОРНА РОБОТА № 2
ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Хід роботи

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

Ознайомтесь з набором даних.

Випишіть у звіт всі 14 ознак з набору даних – їх назви та що вони позначають та вид (числові чи категоріальні).

39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K

(В наборі насправді 15 ознак)

Age (Вік) – Тип: Числова
Workclass (Клас роботи) – Тип: Категоріальна
fnlwgt (Фінальна вага) – Тип: Числова
Education (Освіта) – Тип: Категоріальна
Education-num (Кількість років освіти) – Тип: Числова
Marital-status (Сімейний стан) – Тип: Категоріальна
Occupation (Професія) – Тип: Категоріальна
Relationship (Тип родинних зв'язків) – Тип: Категоріальна
Race (Раса) – Тип: Категоріальна
Sex (Стать) – Тип: Категоріальна
Capital-gain (Капітальний дохід) – Тип: Числова
Capital-loss (Капітальні втрати) – Тип: Числова
Hours-per-week (Кількість годин на тиждень) - Тип: Числова
Native-country (Країна походження) – Тип: Категоріальна
Income (Дохід) – Тип: Категоріальна

Лістинг:

```
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, precision_score,
recall_score, f1_score

input_file = 'income_data.txt'

X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

with open(input_file, 'r') as f:
   for line in f.readlines():
       if count_class1 >= max_datapoints and count_class2 >=
max_datapoints:
           break
       if '?' in line:
           continue

       data = line[:-1].split(', ')

       if data[-1] == '<=50K' and count_class1 < max_datapoints:
           X.append(data)
           Y.append(0)
           count_class1 += 1
       elif data[-1] == '>50K' and count_class2 < max_datapoints:
           X.append(data)
           Y.append(1)
           count_class2 += 1

X = np.array(X)

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

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

```
Y = np.array(Y)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=5)
classifier.fit(X_train, y_train)

y_test_pred = classifier.predict(X_test)

f1 = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

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(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1 Score (v2): {f1 * 100:.2f}%")

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 = np.array([0] * 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 = input_data_encoded.reshape(1, -1)
predicted_class = classifier.predict(input_data_encoded)
print(f"predicted class: {'<=50K' if predicted_class[0] == 0 else
'>50K'}")
```

Результат виконання:

```
[Running] python -u "d:\ztu\KURS4\ai\Lab2\LR_2_task_1.py"
F1 score: 76.01%
Accuracy: 79.56%
Precision: 79.26%
Recall: 79.56%
F1 Score (v2): 75.75%
predicted class: <=50K
```

Рис.1

Зробіть висновок до якого класу належить тестова точка:

Тестова точка належить до класу: **Income (Дохід) - `<=50K`** (річний дохід менше або дорівнює $50,000).

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Для зручнішої роботи з даними та загалом з кодом було написано окремий файл utilities.py, в якому зберігаються фрагменти коду які повторюються та можуть бути винесені в окрему функцію.

Лістинг:

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

def read_and_prepare_data(input_file, max_datapoints=25000):
    X = []
    Y = []
    count_class1 = 0
    count_class2 = 0

    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)
                Y.append(0)
                count_class1 += 1
            elif data[-1] == '>50K' and count_class2 < max_datapoints:
                X.append(data)
                Y.append(1)
                count_class2 += 1

    X = np.array(X)
    print(f"Data successfully read.\nShape of array X: {X.shape}")
    return X, np.array(Y)


def encode_categorical_data(X):
    label_encoder = []
    X_encoded = np.empty(X.shape)
    for i, item in enumerate(X[0]):
        if item.isdigit():
            X_encoded[:, i] = X[:, i]
        else:
            le = preprocessing.LabelEncoder()
            X_encoded[:, i] = le.fit_transform(X[:, i])
            label_encoder.append(le)

    print(f"Encoding completed.\nShape of X after encoding:
{X_encoded.shape}")
    return X_encoded, label_encoder


def split_data(X, Y, test_size=0.2, random_state=5):
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=test_size, random_state=random_state)
    print("Data successfully split into training and testing sets.")
    return X_train, X_test, y_train, y_test


def calculate_and_print_metrics(y_test, y_test_pred):
    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(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Precision: {precision * 100:.2f}%")
    print(f"Recall: {recall * 100:.2f}%")
    print(f"F1 Score: {f1 * 100:.2f}%")
```

LR_2_task_2_1:

```
import numpy as np
from sklearn.svm import SVC
from utilities import read_and_prepare_data, encode_categorical_data,
split_data, calculate_and_print_metrics

input_file = 'income_data.txt'

X, Y = read_and_prepare_data(input_file)

X_encoded, label_encoder = encode_categorical_data(X)

X = X_encoded[:, :-1].astype(int)
Y = np.array(Y)

X_train, X_test, y_train, y_test = split_data(X, Y)

print("Training model with polynomial kernel ('poly')")
classifier = SVC(kernel='poly', degree=2)
classifier.fit(X_train, y_train)

y_test_pred = classifier.predict(X_test)

calculate_and_print_metrics(y_test, y_test_pred)
```

LR_2_task_2_2:

```
import numpy as np
from sklearn.svm import SVC
from utilities import read_and_prepare_data, encode_categorical_data,
split_data, calculate_and_print_metrics

input_file = 'income_data.txt'

X, Y = read_and_prepare_data(input_file)

X_encoded, label_encoder = encode_categorical_data(X)

X = X_encoded[:, :-1].astype(int)
Y = np.array(Y)

X_train, X_test, y_train, y_test = split_data(X, Y)

print("Training model with Gaussian kernel ('rbf')")
classifier = SVC(kernel='rbf')
classifier.fit(X_train, y_train)
```

```
y_test_pred = classifier.predict(X_test)

calculate_and_print_metrics(y_test, y_test_pred)
```

LR_2_task_2_3:

```
import numpy as np
from sklearn.svm import SVC
from utilities import read_and_prepare_data, encode_categorical_data,
split_data, calculate_and_print_metrics

input_file = 'income_data.txt'

X, Y = read_and_prepare_data(input_file)

X_encoded, label_encoder = encode_categorical_data(X)

X = X_encoded[:, :-1].astype(int)
Y = np.array(Y)

X_train, X_test, y_train, y_test = split_data(X, Y)

print("Training model with sigmoid kernel ('sigmoid')")
classifier = SVC(kernel='sigmoid')
classifier.fit(X_train, y_train)

y_test_pred = classifier.predict(X_test)

calculate_and_print_metrics(y_test, y_test_pred)
```

В ході виконання програми було виявлено що якщо використовувати degree=8, то виконання програми буде захмарно довгим:

```
[Running] python -u "d:\ztu\KURS4\ai\Lab2\LR_2_task_2_1.py"
Data successfully read.
Shape of array X: (30162, 15)
Encoding completed.
Shape of X after encoding: (30162, 15)
Data successfully split into training and testing sets.
Training model with polynomial kernel ('poly')

[Done] exited with code=1 in 6073.89 seconds
```

Оптимальним значенням було обрано 2

Результат виконання:

```
Data successfully read.
Shape of array X: (30162, 15)
Encoding completed.
Shape of X after encoding: (30162, 15)
Data successfully split into training and testing sets.
Training model with polynomial kernel ('poly')
Accuracy: 77.39%
Precision: 81.11%
Recall: 77.39%
F1 Score: 70.18%
```

```
Data successfully read.
Shape of array X: (30162, 15)
Encoding completed.
Shape of X after encoding: (30162, 15)
Data successfully split into training and testing sets.
Training model with Gaussian kernel ('rbf')
Accuracy: 78.19%
Precision: 82.82%
Recall: 78.19%
F1 Score: 71.51%
```

```
Data successfully read.
Shape of array X: (30162, 15)
Encoding completed.
Shape of X after encoding: (30162, 15)
Data successfully split into training and testing sets.
Training model with sigmoid kernel ('sigmoid')
Accuracy: 60.47%
Precision: 60.64%
Recall: 60.47%
F1 Score: 60.55%
```

Найкращі результати для класифікації отримані з використанням **гаусового ядра (RBF)**, оскільки воно забезпечило найвищі показники точності, точності класифікації та F1-метрики. Тому, для цієї задачі класифікації рекомендовано використовувати гаусове ядро SVM.
Хоча можливо що при значенні degree=8 результати б були інші.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

КРОК 1. ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ

Лістинг:

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

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

print("Опис набору даних:\n{}".format(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("Значення ознак для перших п'яти
прикладів:\n{}".format(iris_dataset['data'][:5]))

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

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

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names
', 'filename', 'data_module'])
Опис набору даних:
.. _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)
Значення ознак для перших п'яти прикладів:
[[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]]
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 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 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]
PS D:\ztu\KURS4\ai\Lab2>
```

КРОК 2. ВІЗУАЛІЗАЦІЯ ДАНИХ

Лістинг:

```
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(dataset.shape)

print(dataset.head(20))

print(dataset.describe())

print(dataset.groupby('class').size())

dataset.plot(kind='box', subplots=True, layout=(2,2),
sharex=False, sharey=False)
pyplot.show()

dataset.hist()
pyplot.show()

scatter_matrix(dataset)
pyplot.show()
```

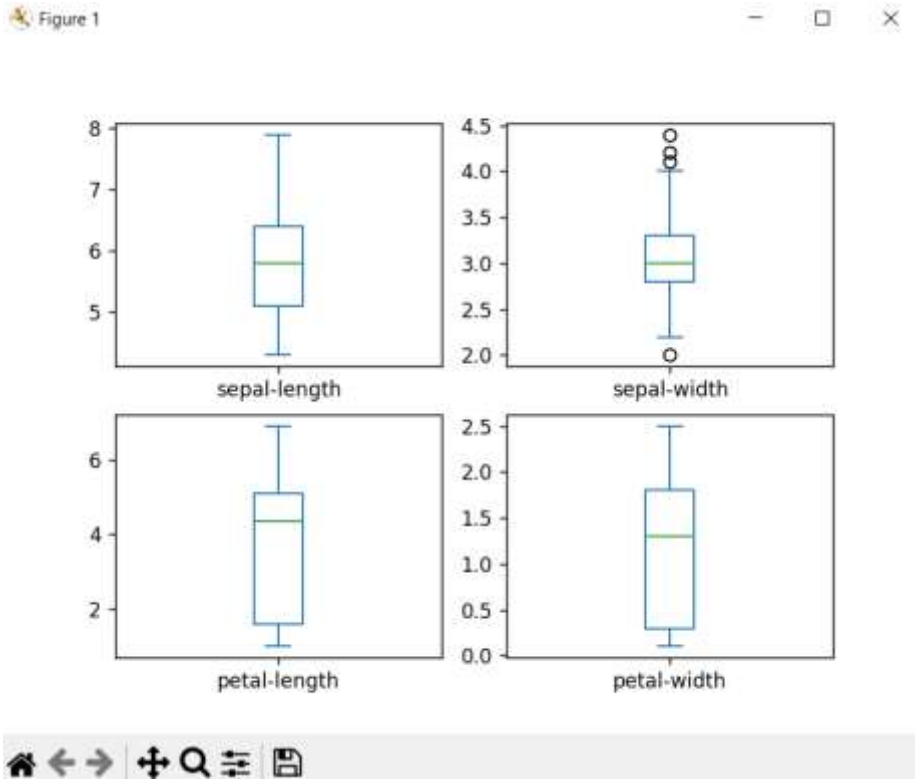


Figure 1
sepal-length
sepal-width
petal-length
petal-width

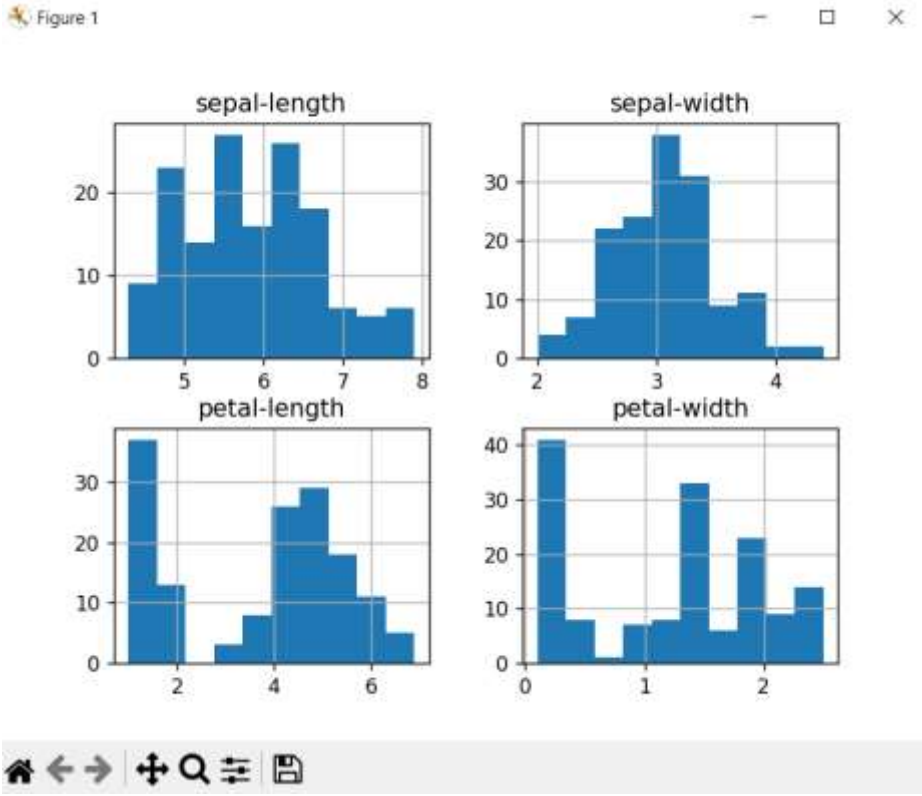
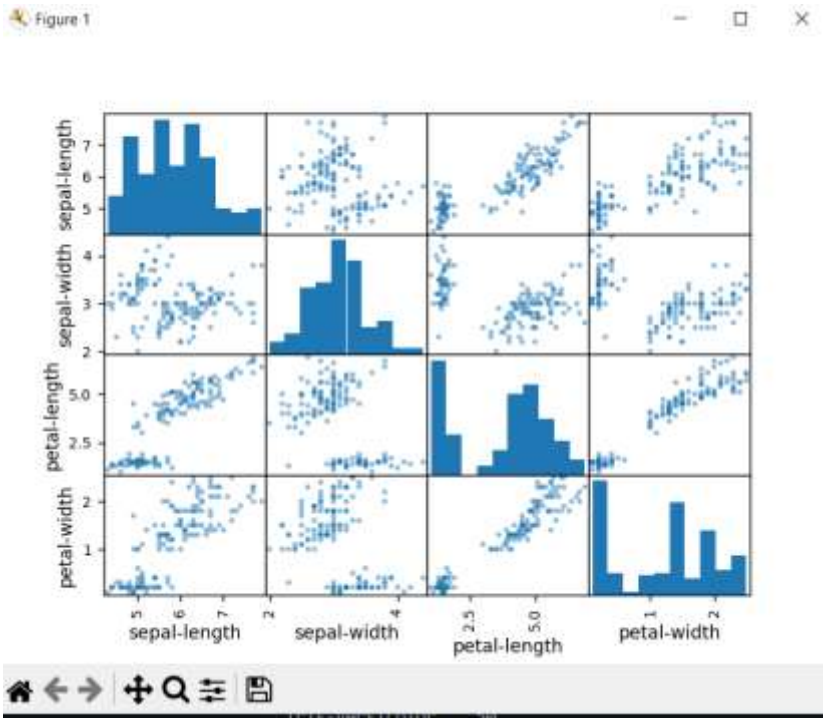
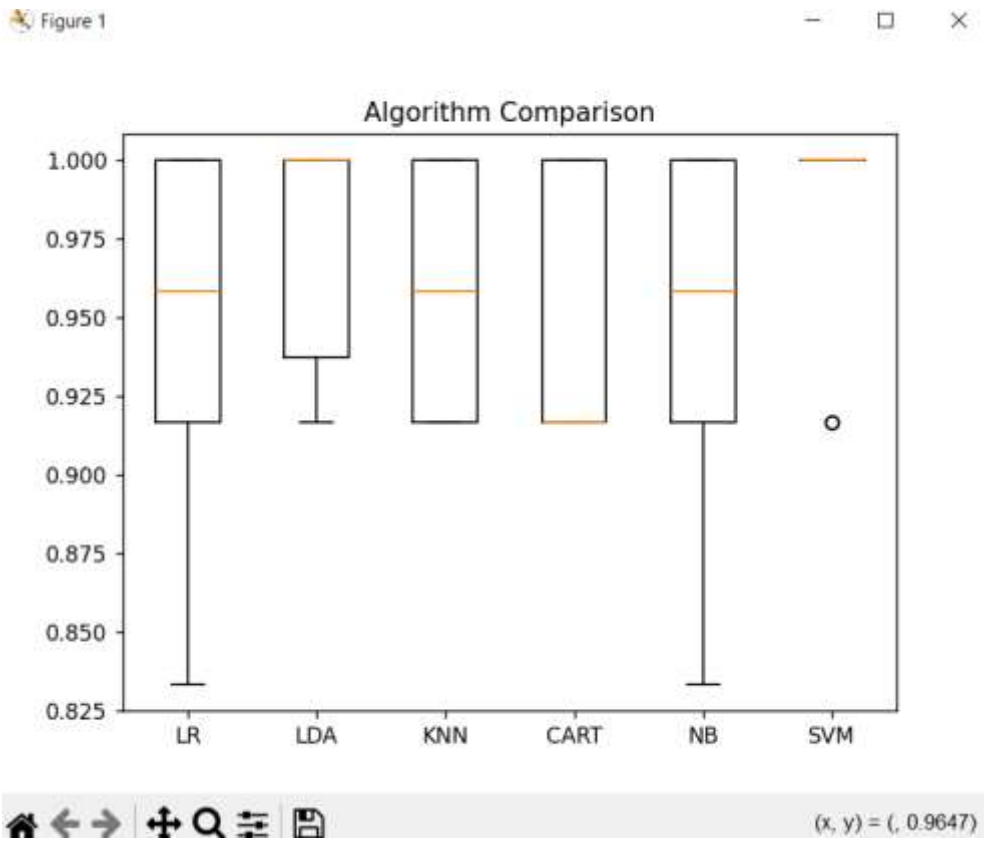


Figure 1
sepal-length
sepal-width
petal-length
petal-width



КРОК 3. СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ

КРОК 4. КЛАСИФІКАЦІЯ (ПОБУДОВА МОДЕЛІ)



```
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)
```

Отримані графіки та результати занесіть у звіт Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим.

Мій вибір це SVM – він має найбільшу точність та найменше відхилення серед всіх.

КРОК 5. ОПТИМІЗАЦІЯ ПАРАМЕТРІВ МОДЕЛІ

КРОК 6. ОТРИМАННЯ ПРОГНОЗУ (ПЕРЕДБАЧЕННЯ НА ТРЕНУВАЛЬНОМУ НАБОРІ)

КРОК 7. ОЦІНКА ЯКОСТІ МОДЕЛІ

```
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
```

Фінальний лістинг:

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
import numpy as np
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)


def print_info():
    print(dataset.shape)
    print(dataset.head(20))
    print(dataset.describe())
    print(dataset.groupby('class').size())

def print_chart():
    dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False,
sharey=False)
    pyplot.show()
    dataset.hist()
    pyplot.show()
    scatter_matrix(dataset)
    pyplot.show()


array = dataset.values

X = array[:,0:4]

Y = array[:,4]

X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y,
test_size=0.20, random_state=1)
```

```
# models = []
# models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
# models.append(('LDA', LinearDiscriminantAnalysis()))
# models.append(('KNN', KNeighborsClassifier()))
# models.append(('CART', DecisionTreeClassifier()))
# models.append(('NB', GaussianNB()))
# models.append(('SVM', SVC(gamma='auto')))

# results = []
# names = []
# for name, model in models:
#     kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
#     cv_results = cross_val_score(model, X_train, Y_train, cv=kfold,
scoring='accuracy')
#     results.append(cv_results)
#     names.append(name)
#     print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))


# pyplot.boxplot(results, tick_labels=names)
# pyplot.title('Algorithm Comparison')
# pyplot.show()


model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)


print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))


X_new = np.array([[5, 2.9, 1, 0.2]])

print("X_new shape: {}".format(X_new.shape))

prediction = model.predict(X_new)
print("prediction: {}".format(prediction))
print("prediction: {}".format(prediction[0]))
```

Вид квітки та решта інформації вказана на рисунку:

```
[Running] python -u "d:\ztu\KURS4\ai\Lab2\LR_2_task_3.py"
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

X_new shape: (1, 4)
prediction: ['Iris-setosa']
prediction: Iris-setosa
```

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг:

```
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, precision_score, recall_score,
f1_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
from matplotlib import pyplot
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold


input_file = 'income_data.txt'
```

```
X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

with open(input_file, 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue

        data = line[:-1].split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            Y.append(0)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            Y.append(1)
            count_class2 += 1

X = np.array(X)

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

X = X_encoded[:, :-1].astype(int)
Y = np.array(Y)

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

models = []
models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
```

```
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

```
LR: 0.791993 (0.005400)
LDA: 0.811637 (0.005701)
KNN: 0.767748 (0.003026)
CART: 0.808032 (0.007684)
NB: 0.789133 (0.006934)
SVM: 0.753119 (0.000378)
```

Можна зробити висновок що конкретно для цієї задачі краще підходить LinearDiscriminantAnalysis, він показав найвищий середній F1 Score серед усіх моделей. Це свідчить про його здатність добре класифікувати дані, враховуючи баланс між точністю та повнотою, а також низький рівень стандартного відхилення.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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 io import BytesIO
from sklearn.metrics import confusion_matrix

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

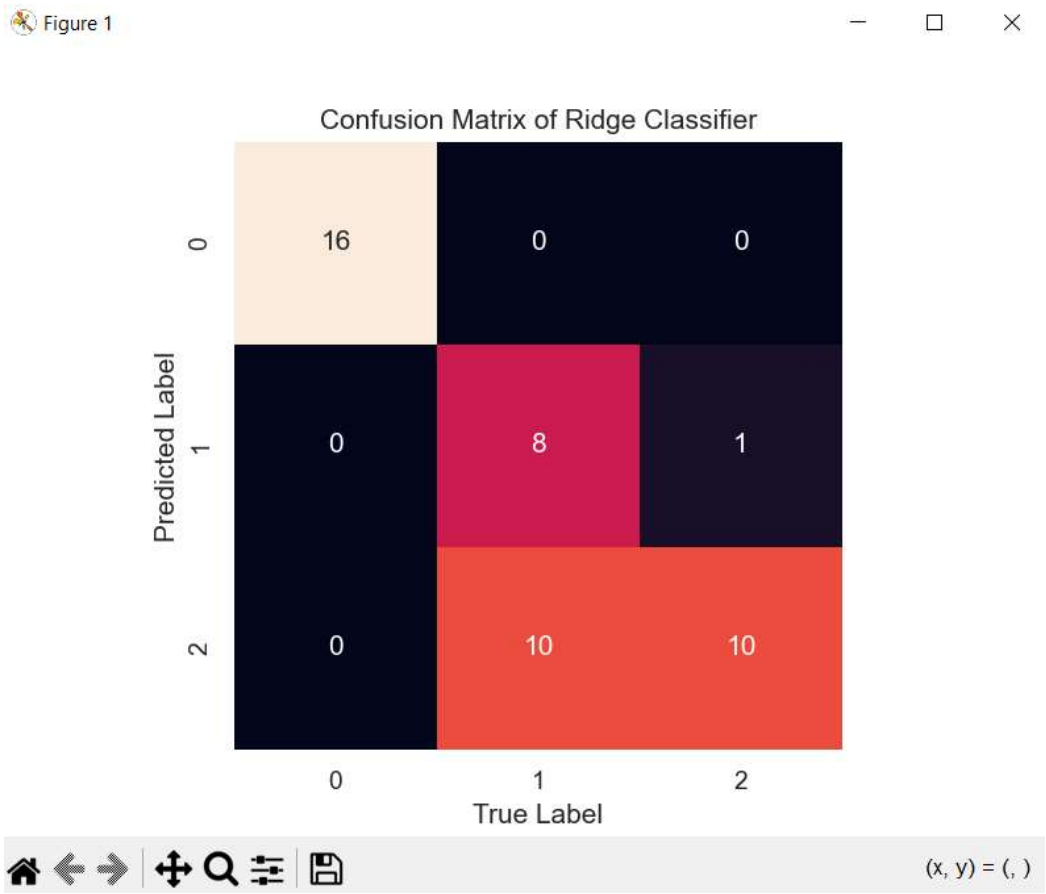
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print('Accuracy:', np.round(metrics.accuracy_score(y_test, y_pred), 4))
print('Precision:', np.round(metrics.precision_score(y_test, y_pred,
average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(y_test, y_pred,
average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(y_test, y_pred,
average='weighted'), 4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(y_test,
y_pred), 4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(y_test,
y_pred), 4))

print('\nClassification Report:\n', metrics.classification_report(y_test,
y_pred))

mat = confusion_matrix(y_test, y_pred)

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 of Ridge Classifier")

plt.savefig("Confusion.jpg")

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

Налаштування класифікатора Ridge:

1. **tol=1e-2**: Параметр tol визначає допустиму похибку для зупинки ітерацій алгоритму. Значення 1e-2 означає, що модель буде зупинятись, якщо зміни в похибці будуть менші за 0.01.

2. **solver='sag'**: solver визначає метод розв’язання задачі оптимізації. Вибір 'sag' (Stochastic Average Gradient) є ітеративним методом, що

ефективно працює на великих наборах даних. Він використовує стохастичний середній градієнт для мінімізації функції втрат.




```
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       0.89      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
```

На матриці плутанини видно наступне:

- Класи:
  - 0: Класифікатор правильно передбачив всі 16 зразків.
  - 1: З 18 зразків 8 було передбачено правильно, 1 зразок було передбачено як клас 0, а 10 зразків передбачено як клас 2.
  - 2: Класифікатор правильно передбачив 10 зразків, 1 зразок було передбачено як клас 1.

Ця матриця плутанини показує, що класифікатор добре працює для класу 0, але має труднощі з відділенням класів 1 та 2.

Коефіцієнт кореляції Метьюза:
Це якість оцінення якості матриці плутанини, Отримане значення 0.6831 показує, що модель є достатньо якісною і демонструє добрий результат класифікування, але при цьому, як вже було зазначено вище, щось їй дається важче, тому результат хоч і не поганий, але точно не хороший.

Коефіцієнта Каппа Коена:
Розраховує ступінь збігів між передбаченнями класифікатора та реальними мітками класів, враховуючи той факт, що частина збігів може бути випадковою. У цьому випадку коефіцієнт Каппа Коена порівнює передбачення моделі Ridge Classifier із істинними класами тестового набору даних Iris. У відповіді досить високий рівень збігів, хоч і не ідеальний.

Git: https://github.com/Alhim616/AI_Labs_Yanushevych