Практична робота № 2

Варіант 13

Попередня обробка та контрольована

Класифікація даних

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних.

Хід роботи:

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

Створіть класифікатор у вигляді машини опорних векторів, призначений для прогнозування меж доходу заданої фізичної особи на основі 14 ознак (атрибутів). Випишіть у звіт всі 14 ознак з набору даних – їх назви та що вони позначають та вид (числові чи категоріальні):

- Вік – чисельна
- Тип працевлаштування – категоріальна
- Рівень освіти – категоріальна
- Безперервність освіти – чисельна
- Сімейний стан – категоріальна
- Професія – категоріальна
- Відносини – категоріальна
- Раса – категоріальна
- Стать – категоріальна
- Приріст капіталу – чисельна
- Витрати – чисельна
- Кількість годин роботи на тиждень – чисельна
- Країна наордження – категоріальна
- Досвід роботи – чисельна

Лістинг файлу task-1.py

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

        data = line[:-1].split(', ')
        if data[-1] == "<=50K" and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

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

X = np.array(X)

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)

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)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)

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

    input_data_encoded = np.array(input_data_encoded)
    predicted_class = classifier.predict([input_data_encoded])
input_data = ['48', 'State-gov', '102628', 'Doctorate', '19', 'Married-civ-
spouse', 'Prof-specialty', 'Husband',
              'Asian-Pac-Islander', 'Male', '0', '0', '40', 'United-States']
predict(input_data)
```

```
Accuracy: 62.64%
Precision: 75.88%
Recall: 62.64%
F1 score: 56.15%
Input data: ['48', 'State-gov', '102628', 'Doctorate', '19', 'Married-civ-spouse', 'Prof-specialty',
Predicted class: <=50K
```

Рис.2.1. task-1.py

Завдання 2.2: Порівняння якості класифікаторів SVM з нелінійними ядрами. Порівнювати між собою будемо поліноміальне, гаусове, сигмоїдальне ядра.

Фрагмент лістингу файлу task-2_1.py (поліноміальне ядро)

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

```
Accuracy: 76.02%
Precision: 64.96%
Recall: 76.02%
F1 score: 66.49%
Input data: ['48', 'State-gov', '102628', 'Doctorate', '19', 'Married-civ-spouse
Predicted class: <=50K
```

Рис.2.2. task-2_1.py

Фрагмент лістингу файлу task-2_2.py (гаусове ядро)

```
classifier = OneVsOneClassifier(SVC(kernel='rbf'))
```

```
Accuracy: 77.48%
Precision: 82.13%
Recall: 77.48%
F1 score: 69.59%
Input data: ['48', 'State-gov', '102628', 'Doctorate', '19', 'Married-civ-spouse',
Predicted class: <=50K
```

Рис.2.2. task-2_2.py

Лістинг файлу task-2_3.py (сигмоїдальне ядро)

```
classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))
```

```
Accuracy: 64.26%
Precision: 63.95%
Recall: 64.26%
F1 score: 64.1%
Input data: ['48', 'State-gov', '102628', 'Doctorate', '19', 'Married-civ-spouse', 'Prof-specialty',
Predicted class: <=50K
```

Рис.2.2. task-2_3.py

Завдання 2.3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів. Необхідно класифікувати сорти ірисів за деякими їх характеристиками: довжина та ширина пелюсток, а також довжина та ширина чашолистків.

Також, в наявності є вимірювання цих же характеристик ірисів, які раніше дозволили досвідченому експерту віднести їх до сортів: setosa, versicolor і virginica.

Лістинг файлу task-2_4.py

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

print(f"Iris_dataset keys: \n{iris_dataset.keys()}")
print(iris_dataset['DESCR'][:193] + "\n...")

print(f"Response names: {iris_dataset['target_names']}")
print(f"Feature names: {iris_dataset['feature_names']}")
print(f"Data type: {type(iris_dataset['data'])}")
print(f"Data size: {iris_dataset['data'].shape}")
print(f"The first five lines of data:\n{iris_dataset['data'][:5]}")
print(f"Response array type: {type(iris_dataset['target'])}")
print(f"Size of response array: {iris_dataset['target'].shape}")
print(f"Answers:\n{iris_dataset['target']}")
```

```
Answers:
[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 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]
```

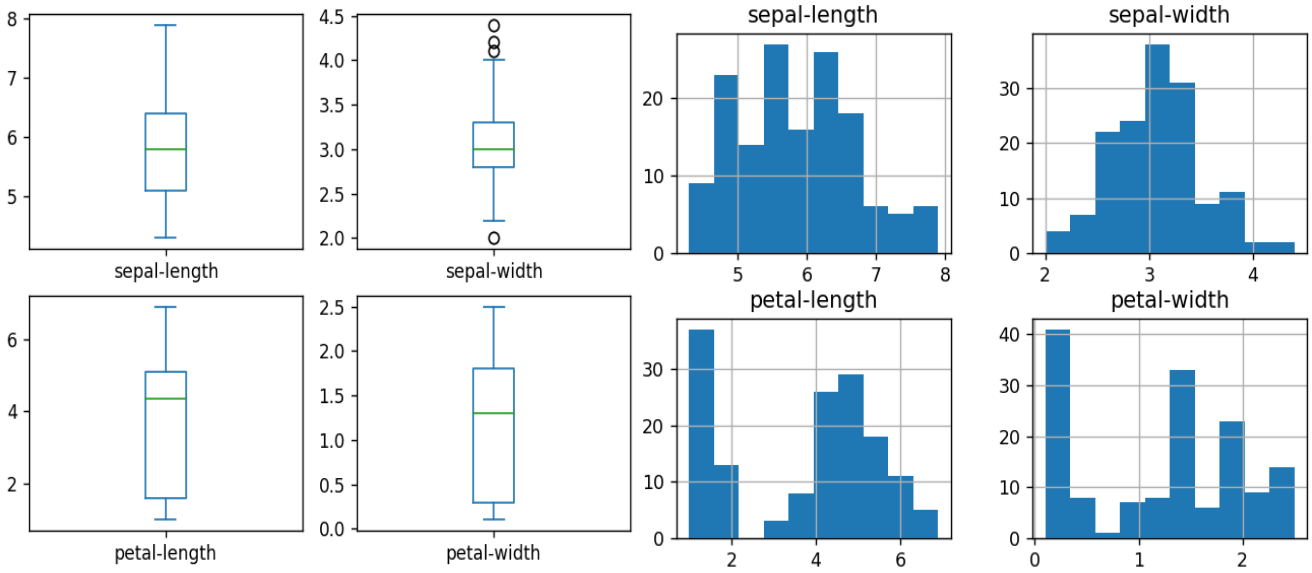
```
Iris_dataset keys:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
Response names: ['setosa' 'versicolor' 'virginica']
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Data type: <class 'numpy.ndarray'>
Data size: (150, 4)
The first five lines of 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]]
Response array type: <class 'numpy.ndarray'>
Size of response array: (150,)
```

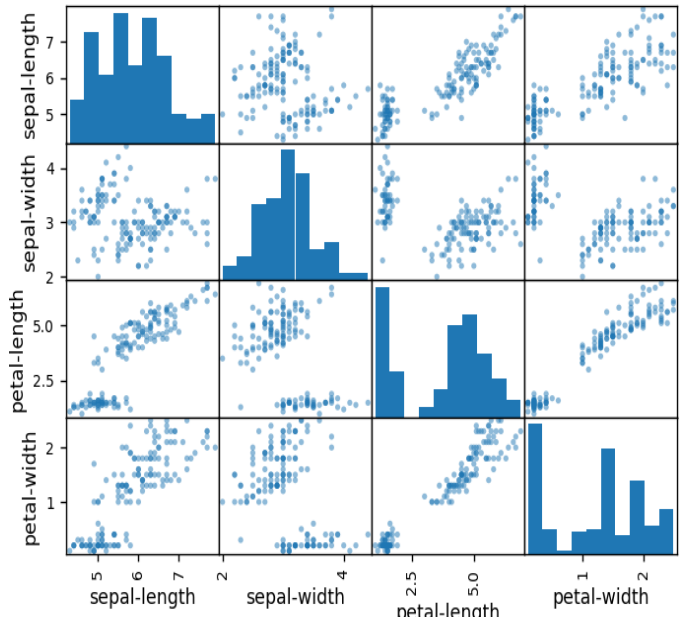
Рис.2.2. task-2_4.py

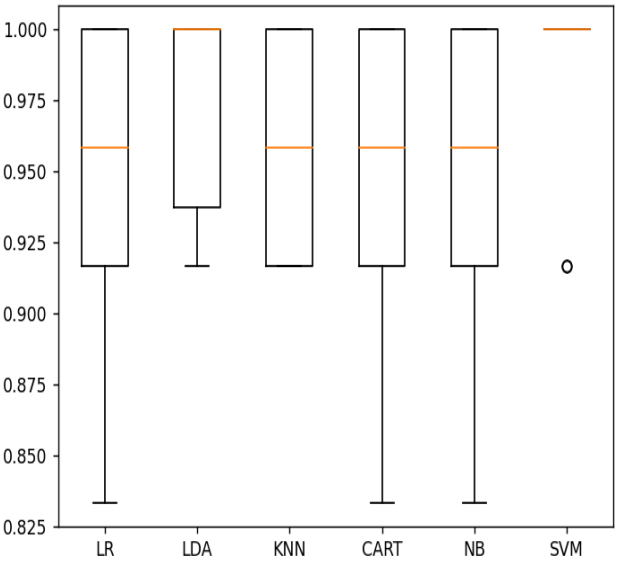
Лістинг файлу task-2_5.py

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix,
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)
plt.show()

dataset.hist()
plt.show()
```

```
scatter_matrix(dataset)
plt.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 = [
    ('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    ('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(f'{name}: {cv_results.mean()} ({cv_results.std()})')

plt.boxplot(results, labels=names)
plt.title('Алгоритми порівняння')
plt.show()

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

X_new = [[5.0, 3.6, 1.3, 0.25], [5.9, 3.0, 5.1, 1.8], [6.3, 3.3, 6.0, 2.5], [5.8,
2.7, 5.1, 1.9], [5.1, 3.5, 1.4, 0.2]]
predictions = model.predict(X_new)
print(f"X_new: {X_new}\nPredictions: {predictions}")
```








```
C:\Python311\python.exe C:/Users/Ola/Documents/stud/4/AI_Python/Lr_2/lab-2/task-2_5.py
(150, 5)
    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
```

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
LR: 0.9416666666666667 (0.06508541396588878)
LDA: 0.975 (0.03818813079129868)
KNN: 0.9583333333333333 (0.04166666666666669)
CART: 0.9416666666666667 (0.05335936864527374)
NB: 0.95 (0.05527707983925667)
SVM: 0.9833333333333332 (0.03333333333333335)
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: [[5.0, 3.6, 1.3, 0.25], [5.9, 3.0, 5.1, 1.8], [6.3, 3.3, 6.0, 2.5], [5.8, 2.7, 5.1, 1.9], [5.1, 3.5, 1.4, 0.2]]
Predictions: ['Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa']
```

Рис.2.2. task-2_5.py

https://github.com/avrorilka/AI_Python

Висновки: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python ми дослідити попередню обробку та класифікацію даних.

Класифікацію та кінцевий аналіз даних ми проводили на прикладах класифікатору у вигляді машини опорних векторів, призначений для прогнозування меж доходу заданої фізичної особи на основі 14 атрибутів.

Після чого провели перехресну перевірку, розбивши дані на навчальний та тестовий набори у пропорції 80/20. А для порівняння якостей класифікаторів SVM з нелінійними ядрами ми порівнювали між собою поліноміальне, гаусове та сигмоїдальне ядра.