Лабораторна робота №2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Мета роботи: 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).

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

№	Назва	Позначають	Вид
1	Age	Вік	Integer (Числові)
2	Workclass	Дохід	Categorical (Категоріальні)
3	Fnlwgt	-	Integer (Числові)
4	Education	Освіта	Categorical (Категоріальні)
5	Education-num	Рівень освіти	Integer (Числові)
6	Marital-status	Сімейний стан	Categorical (Категоріальні)
7	Occupation	Професія	Categorical (Категоріальні)
8	Relationship	Відносини	Categorical (Категоріальні)
9	Race	Раса	Categorical (Категоріальні)
10	Sex	Стать	Binary (Категоріальні)
11	Capital-gain	Приріст капіталу	Integer (Числові)
12	Capital-loss	Збиток капіталу	Integer (Числові)
13	Hours-per-week	Години на тиждень	Integer (Числові)
14	Native-country	Батьківщина	Categorical (Категоріальні)
15	Income	Дохід	Binary (Числові)

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, cross_val_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)

# Перетворення рядкових даних на числові
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)

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, dual=False,
max_iter=10000))
classifier.fit(X_train, y_train)

y_test_pred = classifier.predict(X_test)

f1 = cross_val_score(classifier, X, y, scoring='f1_macro', cv=3)
print("F1 score: " + str(round(100*f1.mean(), 2)) + "%")

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Accuracy score: " + str(round(100 * accuracy.mean(), 2)) + "%")
```

```
precision = cross_val_score(classifier, X, y, scoring='precision', cv=3)
print("Precision score: " + str(round(100 * precision.mean(), 2)) + "%")

recall = cross_val_score(classifier, X, y, scoring='recall', cv=3)
print("Recall score: " + str(round(100 * recall.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]])[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])
```

Результат виконання програми:

```
C:\Users\Димка\AppData\Loca
F1 score: 64.2%
Accuracy score: 79.66%
Precision score: 74.85%
Recall score: 28.01%
<=50K
```

Рис. 2.1.1 – Результат виконання завдання.

***Висновок*:**

Ця тестова точка відноситься до групи людей, заробляння яких становить менше або дорівнює 50 тисячам.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Лістинг програми (з поліноміальним ядром):

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
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)
            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)

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

classifier = SVC(kernel="poly", degree=8)
classifier.fit(X_train_scaled, y_train)
y_test_pred = classifier.predict(X_test_scaled)

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

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

Результат виконання програми:

```
C:\Users\Димка\AppData\Loca
Accuracy score: 77.94%
Precision score: 67.62%
Recall score: 26.15%
F1 score: 37.72%
```

Рис. 2.2.1 – Результат виконання завдання (1).

Лістинг програми (з гаусовим ядром):

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
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)
            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)
```

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

classifier = SVC(kernel="rbf")
classifier.fit(X_train_scaled, y_train)
y_test_pred = classifier.predict(X_test_scaled)

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

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

Результат виконання програми:

```
C:\Users\Димка\AppData\Loc
Accuracy score: 83.34%
Precision score: 73.67%
Recall score: 54.12%
F1 score: 62.40%
```

Рис. 2.2.2 – Результат виконання завдання (2).

Лістинг програми (з сигмоїдальним ядром):

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
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)
            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)

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

classifier = SVC(kernel="sigmoid")
classifier.fit(X_train_scaled, y_train)
y_test_pred = classifier.predict(X_test_scaled)

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

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

Результат виконання програми:

```
C:\Users\Димка\AppData\Loc
Accuracy score: 75.25%
Precision score: 51.60%
Recall score: 50.16%
F1 score: 50.87%
```

Рис. 2.2.3 – Результат виконання завдання (3).

Висновок:

Найбільш точним і дбайливим класифікатором є нелінійний SVM з гаусовим ядром, але що стосується повноти, найкращим виявився нелінійний SVM з

поліноміальним ядром. Загалом, для виконання завдання найкращим класифікатором є той, що використовує гаусове ядро.

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(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 п'яти прикладів: {}".format(iris_dataset['data'][:5]))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

Результат виконання програми:

```
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, pre
...
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 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'>
```

Рис. 2.3.1 – Результат виконання завдання (1).

```
Форма масиву data: (150, 4)
Значення ознак для 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 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 2]
```

Рис. 2.3.2 – Результат виконання завдання (2).

Лістинг програми:

```
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
import numpy as np

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()

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, 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("форма масиву X_new: {}".format(X_new.shape))

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

Результат виконання програми:

```
(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
```

Рис. 2.3.3 – Результат виконання завдання (1).

```
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: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.055277)
NB: 0.950000 (0.055277)
```

Рис. 2.3.4 – Результат виконання завдання (2).

```
SVM: 0.983333 (0.033333)
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

форма масиву X_new: (1, 4)
Прогноз: ['Iris-setosa']
Спрогнозована мітка: Iris-setosa
```

Рис. 2.3.5 – Результат виконання завдання (3).

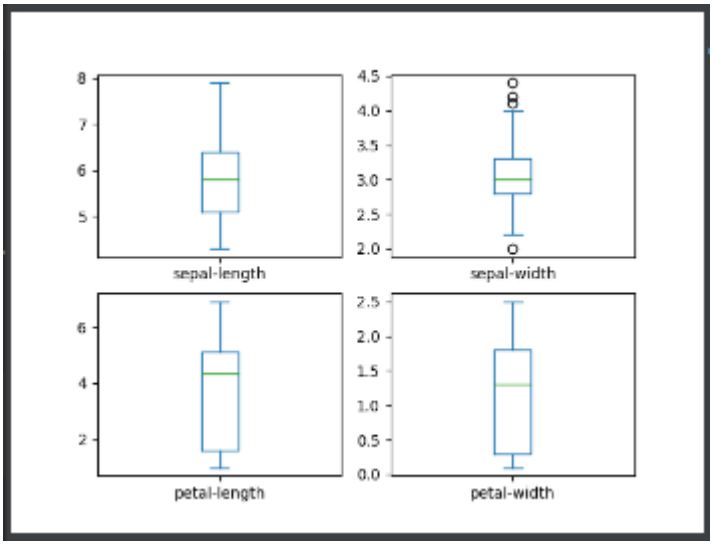

Рис. 2.3.6 – Результат виконання завдання (3).

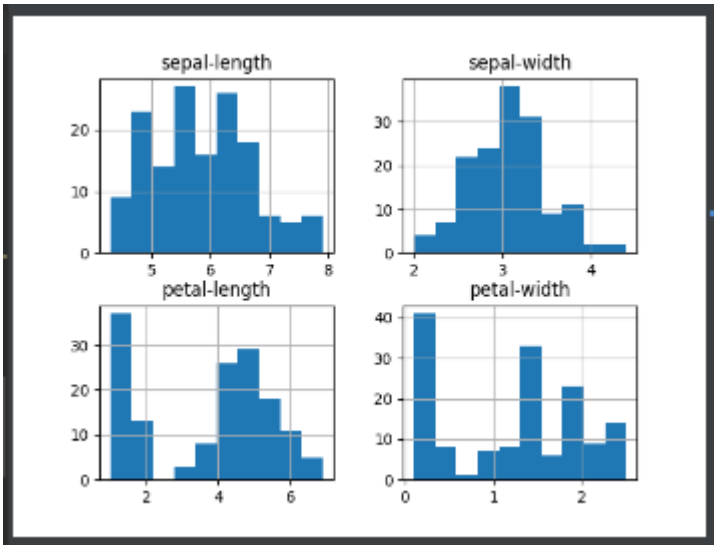

Рис. 2.3.7 – Результат виконання завдання (4).

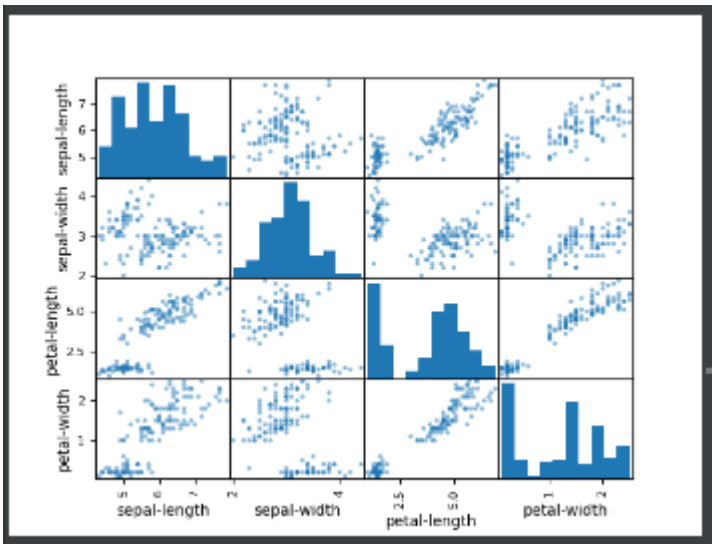

Рис. 2.3.8 – Результат виконання завдання (5).

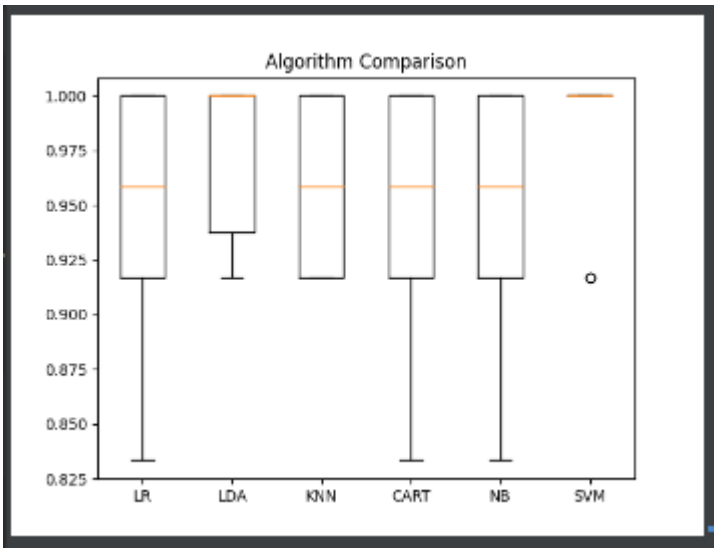

Рис. 2.3.9 – Результат виконання завдання (6).

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

- *Обраний метод класифікації - "SVM" (Support Vector Machine) - вважаю найкращим через його здатність працювати добре навіть у випадку складних нелінійних вибірок із великою кількістю ознак. В даному випадку, модель "SVC" (SVM з ядром "auto") показала високу точність під час перевірки на тестовій вибірці та має потенціал для роботи з новими даними.*

Висновок:

Квітка належить до класу Setosa. Вдалося досягти 0.966…7 показника якості.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1.

Лістинг програми:

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
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
import numpy as np

input_file = 'income_data.txt'
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.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)

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)

X_train, X_validation, y_train, y_validation = train_test_split(X, y,
test_size=0.2, 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('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

Результат виконання програми:

```
LR: 0.793070 (0.006099)
LDA: 0.812176 (0.003802)
KNN: 0.766961 (0.006871)
CART: 0.804343 (0.008502)
NB: 0.789796 (0.004791)
```

Рис. 2.4.1 – Результат виконання завдання.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge.

Лістинг програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from io import BytesIO
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)

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

Результат виконання програми:

```
C:\Users\Димка\AppData\Local\Programs\Python\Python39\python.exe "D:/4
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       0.44      0.89      0.59         9
           2       0.91      0.50      0.65        20

    accuracy                           0.76        45
   macro avg       0.78      0.80      0.75        45
weighted avg       0.85      0.76      0.76        45
```

Рис. 2.5.1 – Результат виконання завдання (1).

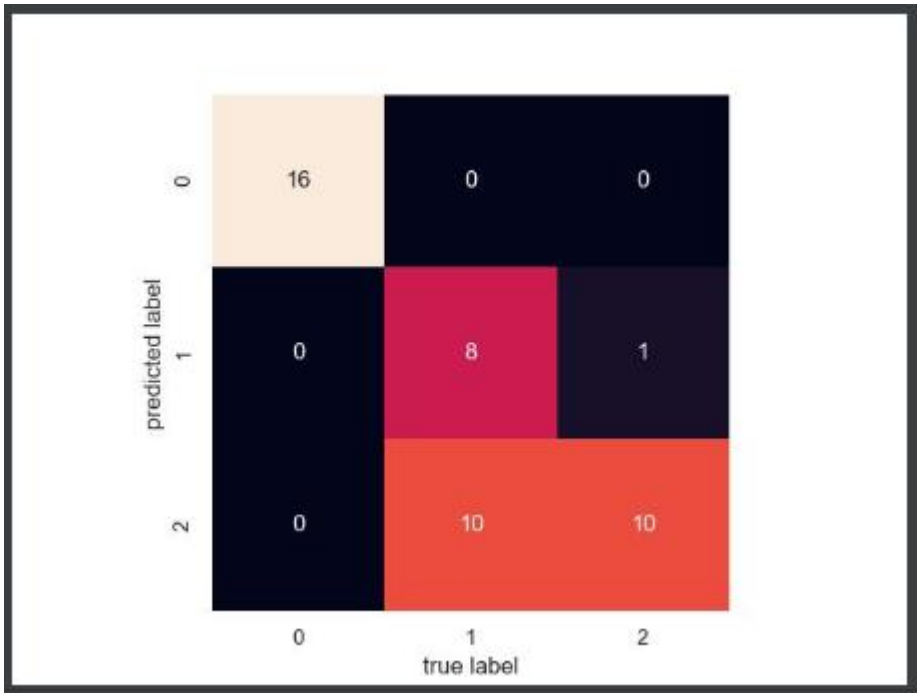

Рис. 2.5.2 – Результат виконання завдання (2).

Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають.

- *В класифікатора Ridge використовані наступні налаштування: параметр tol встановлено на значення 1e-2 (це допуск для зупинки оптимізаційного алгоритму), і як рішальник використовується "sag" (Stochastic Average Gradient Descent), що є одним із методів оптимізації для RidgeClassifier.*

Опишіть які показники якості використовуються та їх отримані результати. Вставте у звіт та поясніть зображення Confusion.jpg

- *Accuracy, Precision, Recall, F1 score, Cohen Kappa Score, Matthews Correlation Coefficient.*
- *Зображення "Confusion.jpg" відображає матрицю помилок, де по діагоналі показано правильно класифіковані приклади для кожного класу, а поза діагоналлю - помилкові класифікації.*

Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.

- *Коефіцієнт Коена Каппа і коефіцієнт кореляції Метьюза використовуються для вимірювання узгодженості між фактичними та передбаченими класами. Коефіцієнт Коена Каппа враховує випадковість та відображає, наскільки збігаються фактичні та передбачені класи, враховуючи інтеркориговану випадковість. Коефіцієнт кореляції Метьюза також враховує випадковість, і він вимірює ступінь кореляції між спостереженнями та передбаченнями, де 1 вказує на ідеальну узгодженість, -1 на повну протилежність, а 0 - на випадкову узгодженість. Ці коефіцієнти оцінюють ступінь надійності класифікатора.*

Посилання на репозиторій: **https://github.com/GrunytskyDmytro/Lab2_AI**

Висновок по лабораторній роботі: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.