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

GitHub репозиторій: **https://github.com/SMTH666/OAI-Lab**

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

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
# 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
        elif 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)
```

```
# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0, dual=False,
max_iter=10000))
# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
# Навчання класифікатора
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
# Обчислення F-міри для SVM-класифікатора
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_test_pred, average="weighted")
print("F1 score: " + str(round(100 * f1, 2)) + "%")
input_data = ["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-
spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([in-
put_data[i]])[0])
        count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
# Використання класифікатора для кодованої точки даних
predicted_class = classifier.predict(input_data_encoded)
predicted_label = label_encoder[-1].inverse_transform(predicted_class)[0]
print(predicted_label)
# Обчислення акуратності
accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:" + str(round(100 * accuracy, 2)) + "%")
# Обчислення точності
precision = precision_score(y_test, y_test_pred, average="weighted")
print("Precision:" + str(round(100 * precision, 2)) + "%")
# Обчислення повноти
recall = recall_score(y_test, y_test_pred, average="weighted")
print("Recall:" + str(round(100 * recall, 2)) + "%")
```

Результат виконання програми:

```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\main.py
F1 score: 75.75%
>50K
Accuracy:79.56%
Precision:79.26%
Recall:79.56%

Process finished with exit code 0
```

Висновок: Тестова точка була класифікована як ">50K". Це означає, що модель передбачає, що вище 50 000 доларів на рік заробляє ця людина

Завдання 2.2: Порівняння якості класифікаторів SVM з нелінійними ядрами

Лістинг програми LR_2_task_2_1.py:

```
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
from sklearn.svm import SVC

# 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
        elif 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)
# Створення SVM-класифікатора з поліноміальним ядром
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, random_state=0))
# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
# Навчання класифікатора
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
# Обчислення F-міри для SVM-класифікатора
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_test_pred, average="weighted")
print("F1 score: " + str(round(100 * f1, 2)) + "%")
input_data = ["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-
spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([in-
put_data[i]])[0])
```

```
        count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
# Використання класифікатора для кодованої точки даних
predicted_class = classifier.predict(input_data_encoded)
predicted_label = label_encoder[-1].inverse_transform(predicted_class)[0]
print(predicted_label)
# Обчислення акуратності
accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:" + str(round(100 * accuracy, 2)) + "%")
# Обчислення точності
precision = precision_score(y_test, y_test_pred, average="weighted")
print("Precision:" + str(round(100 * precision, 2)) + "%")
# Обчислення повноти
recall = recall_score(y_test, y_test_pred, average="weighted")
print("Recall:" + str(round(100 * recall, 2)) + "%")
```

Результати виконання:

Класифікатор з поліноміальним ядром не працює

```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\LR_2_task_1.py
```

Лістинг програми LR_2_task_2_2.py:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_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
        elif 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)
# Створення SVM-класифікатора
classifier = OneVsOneClassifier(SVC(kernel='rbf', random_state=0))
# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
# Навчання класифікатора
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
# Обчислення F-міри для SVM-класифікатора
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_test_pred, average="weighted")
print("F1 score: " + str(round(100 * f1, 2)) + "%")
input_data = ["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-
spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([in-
put_data[i]])[0])
        count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
# Використання класифікатора для кодованої точки даних
predicted_class = classifier.predict(input_data_encoded)
predicted_label = label_encoder[-1].inverse_transform(predicted_class)[0]
print(predicted_label)
# Обчислення акуратності
accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:" + str(round(100 * accuracy, 2)) + "%")
# Обчислення точності
precision = precision_score(y_test, y_test_pred, average="weighted")
print("Precision:" + str(round(100 * precision, 2)) + "%")
# Обчислення повноти
recall = recall_score(y_test, y_test_pred, average="weighted")
print("Recall:" + str(round(100 * recall, 2)) + "%")
```

Результати виконання програми:

```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\LR_2_task_2.py
F1 score: 71.51%
>50K
Accuracy:78.19%
Precision:82.82%
Recall:78.19%

Process finished with exit code 0
```

Лістинг програми LR_2_task_2_3.py:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_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
        elif 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)
# Створення SVM-класифікатора
classifier = OneVsOneClassifier(SVC(kernel='sigmoid', random_state=0))
# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
# Навчання класифікатора
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
# Обчислення F-міри для SVM-класифікатора
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_test_pred, average="weighted")
print("F1 score: " + str(round(100 * f1, 2)) + "%")
input_data = ["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-
spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([in-
put_data[i]])[0])
        count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
# Використання класифікатора для кодованої точки даних
predicted_class = classifier.predict(input_data_encoded)
predicted_label = label_encoder[-1].inverse_transform(predicted_class)[0]
print(predicted_label)
# Обчислення акуратності
accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:" + str(round(100 * accuracy, 2)) + "%")
# Обчислення точності
precision = precision_score(y_test, y_test_pred, average="weighted")
print("Precision:" + str(round(100 * precision, 2)) + "%")
# Обчислення повноти
recall = recall_score(y_test, y_test_pred, average="weighted")
print("Recall:" + str(round(100 * recall, 2)) + "%")
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\LR_2_task_3.py
F1 score: 60.55%
>50K
Accuracy:60.47%
Precision:60.64%
Recall:60.47%

Process finished with exit code 0
```

Висновок: Найрезультативнішим по всім показникам є класифікатор з гаусовим ядром.

Завдання 2.3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Лістинг програми:

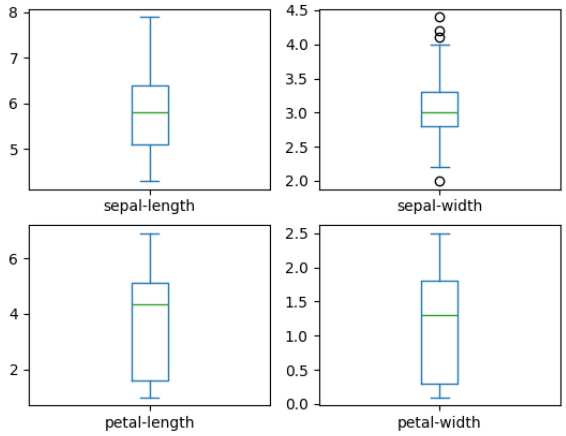
```
import numpy as np
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics 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)
```

```
# shape
print(dataset.shape)
# Зріз даних head
print(dataset.head(20))
# Стастичні зведення методом describe
print(dataset.describe())
# Розподіл за атрибутом class
print(dataset.groupby('class').size())

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

# Гістограма розподілу атрибутів датасета
dataset.hist()
pyplot.show()

#Матриця діаграм розсіювання
scatter_matrix(dataset)
pyplot.show()

# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
# Вибір перших 4-х стовпців
X = array[:, 0:4]

# Вибір 5-го стовпця
y = array[:, 4]
X_train, X_validation, Y_train, Y_validation = train_test_split(
    X, y, test_size=0.20, random_state=1)

# Завантажуємо алгоритми моделі
models = []
models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# оцінюємо модель на кожній ітерації
results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1,shuffle=True)
    cv_results = cross_val_score(model, X_train, Y_train,cv=kfold, scoring='accu-
racy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(),
    cv_results.std()))

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

# Створюємо прогноз на контрольній вибірці
model = SVC(gamma="auto")
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

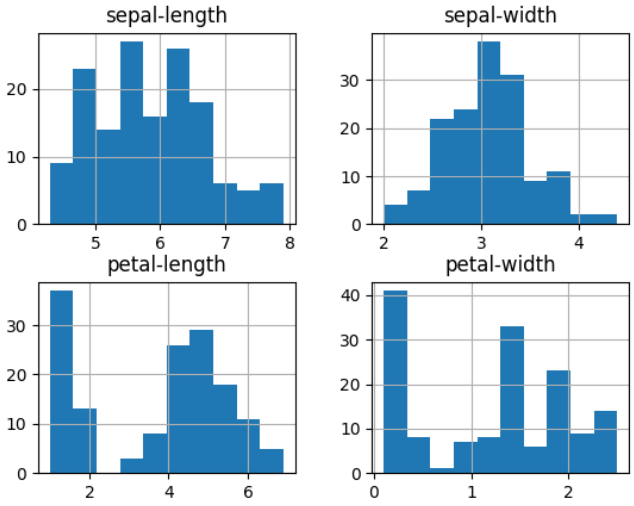
```
# Оцінюємо прогноз
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

#Отримуємо прогноз
X_new = np.array([[5.0, 2.9, 1.0, 0.2]])

X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма массива X_new: {}".format(X_new.shape))
prediction = model.predict(X_new)
print("Прогноз: {}".format(prediction))
print("Спрогнозована мітка: {}".format(prediction[0]))
```

Результат виконання програми:

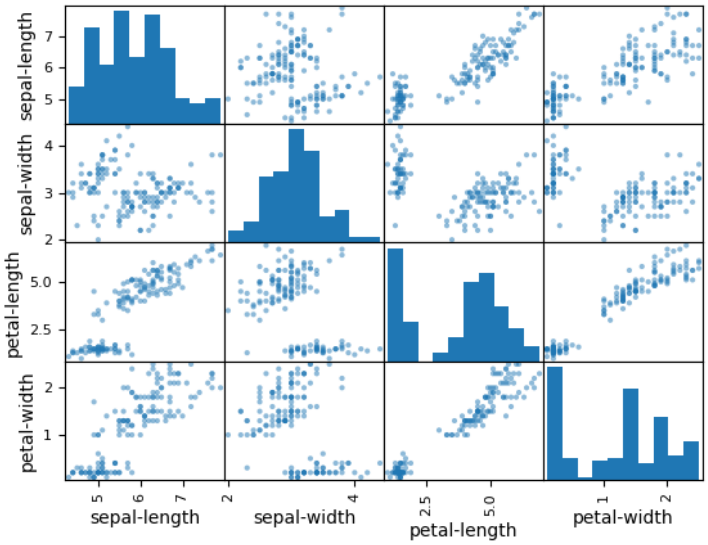
```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\LR_2_task_3.py
(150, 5)
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           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: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.055277)
NB: 0.950000 (0.055277)
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 массива X_new: (1, 4)
Прогноз: ['Iris-setosa']
Спрогнозована мітка: Iris-setosa

Process finished with exit code 0
```



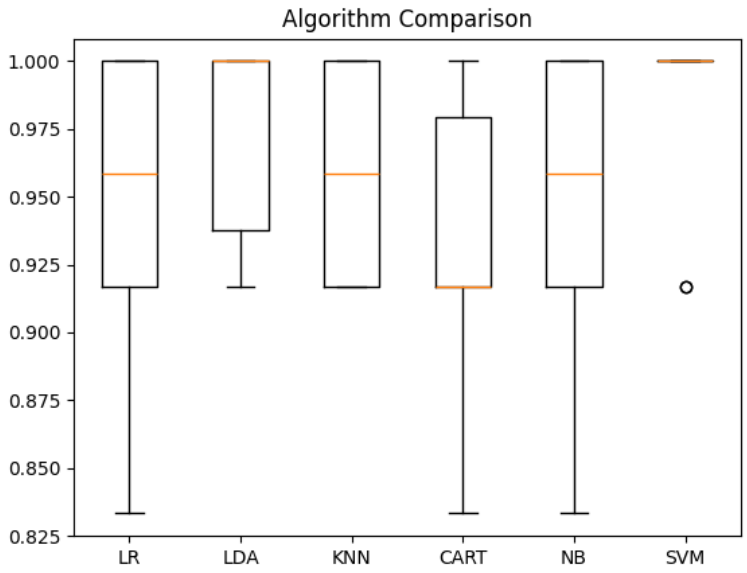
Діаграма розмаху



Гістограма розподілу атрибутів датасета



Матриця діаграм розсіювання



Алгоритм порівняння

Завдання 2.4

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
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

input_file = 'income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000
#Відкриємо файл і прочитаємо рядки.
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
        elif 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, ran-
dom_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', max_iter=10000)))

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='accura-
cy')
  results.append(cv_results)
  names.append(name)
  print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\LR_2_task_4.py
LR: 0.791993 (0.005400)
LDA: 0.811637 (0.005701)
KNN: 0.767748 (0.003026)
CART: 0.807700 (0.006968)
NB: 0.789133 (0.006934)
C:\Users\toxa1\PycharmProjects\lab02\venv\Lib\site-packages\sklearn\svm\_base.py:297: ConvergenceWarning: Solver terminated early (max_iter=10000).  Consider
  warnings.warn(
```

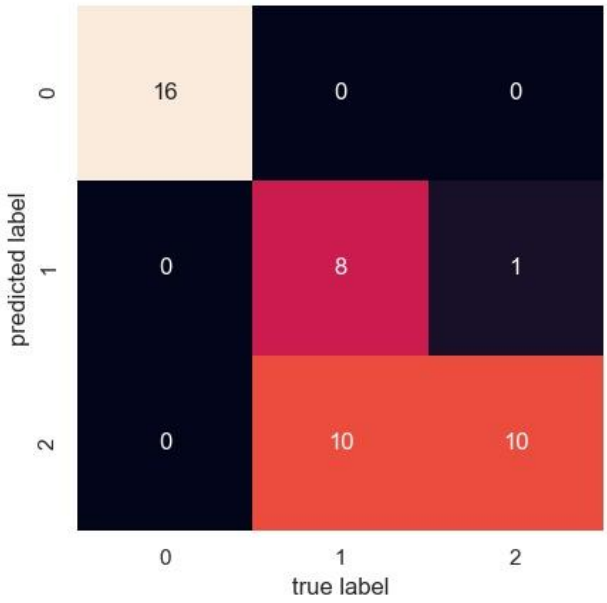
Завдання 2.5

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, ran-
dom_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,aver-
age='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, aver-
age='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_re-
port(y_pred,y_test))
mat = confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.savefig("Confusion.jpg")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format="svg")
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab02\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab02\LR_2_task_5.py
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


Process finished with exit code 0
```



Файл Confusion

1. **Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають.**
   В класифікаторі Ridge були використані налаштування точності (tol=1e-2) та розв'язник(solver="sag").
2. **Опишіть які показники якості використовуються та їх отримані результати**
   Показники якості– акуратність, точність, повнота, коефіцієнт Коена Каппа, коефіцієнт кореляції Метьюза.
3. **Вставте у звіт та поясніть зображення Confusion.jpg**
   На зображені показана матриця confusion, як skicit-learn може навчатися класифікувати.
4. **Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.**

Коефіцієнт Коена Каппа – це статистичний показник, який використовується для вимірювання згоди або узгодженості між двома оцінювачами або системами оцінювання. Зазвичай використовується у контексті оцінки точності класифікаційних моделей, особливо в задачах класифікації, де важлива не тільки точність, але і узгодженість між прогнозами. В даному випадку він показує істотну згоду.

Коефіцієнт кореляції Метьюза - це інший статистичний показник, який використовується для оцінки якості класифікаційних моделей, особливо в задачах бінарної класифікації (тобто, коли є два класи - позитивний і негативний). Незважаючи на високу точність, акуратність і повноту в нашому випадку, коефіцієнт кореляції Метьюза становить 0.6831. Це вказує на високу оцінку цього коефіцієнта в ситуаціях, коли класифікатор успішно розпізнає як негативні, так і позитивні значення.

Висновки: на даній лабораторній ми, використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчились їх порівнювати.