ЛАБОРАТОРНА РОБОТА № 2

Тема: ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

Хід роботи:

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

14 ознак з набору данних(із прикладом):

1. 39 - числова ознака: вік.
2. State-gov - категоріальна ознака: тип зайнятості (у цьому випадку, державна служба).
3. 77516 - числова ознака: унікальний ідентифікатор або номер.
4. Bachelors - категоріальна ознака: рівень освіти (у цьому випадку, бакалаврська).
5. 13 - числова ознака: кількість років навчання (як частина рівня освіти).
6. Never-married - категоріальна ознака: сімейний стан (у цьому випадку, ніколи не одружений/незаміжня).
7. Adm-clerical - категоріальна ознака: професія або робоча посада (у цьому випадку, адміністративний клерк).
8. Not-in-family - категоріальна ознака: статус в сім'ї (у цьому випадку, не в родині).
9. White - категоріальна ознака: раса (у цьому випадку, білий).
10. Male - категоріальна ознака: стать (чоловік).
11. 0 - числова ознака: кількість власних дітей.
12. 40 - числова ознака: кількість годин роботи на тиждень.
13. United-States - категоріальна ознака: країна проживання (у цьому випадку, Сполучені Штати Америки).
14. <=50K - категоріальна ознака: цільова змінна, яка може позначати дохід (у цьому випадку, менше або рівно 50 тисяч доларів на рік).
15.

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,
random_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)) + "%")

# Передбачення результату для тестової точки даних
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([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 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\1\AppData\Local\Prog
F1 score: 75.75%
>50K
Accuracy:79.56%
Precision:79.26%
Recall:79.56%
```

Рис.1 Результат виконання програми

Тестова точка належить до класу >50K

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

З поліноміальним ядром:

```
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
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C': [0.1, 1, 10],
    'degree': [2, 3, 4],
    'coef0': [0, 1, 2]
}

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', C = 1.0, degree = 8, coef0 =
1))
```

```
# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_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([input_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)) + "%")
```

З гаусовим ядром:

```
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,
random_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([input_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:

```
F1 score: 71.51%
>50K
Accuracy:78.19%
Precision:82.82%
Recall:78.19%
```

Рис.3 Результат з Гаусовим ядром

З сигмоїдальним ядром:

```
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,
random_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([input_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:

```
F1 score: 60.55%
>50K
Accuracy:60.47%
Precision:60.64%
Recall:60.47%
```

Рис.4 Результат з сигмоїдальним ядром

1. **Поліноміальне ядро**: Використовується поліноміальне ядро, коли вважаємо, що взаємозв'язки між ознаками можуть бути аппроксимовані поліномами
2. **Гаусове ядро (RBF)**: Гаусове ядро зазвичай є добрим вибором за замовчуванням, оскільки воно добре працює з різними типами даних і здатне моделювати складні нелинійні залежності.
3. **Сигмоїдальне ядро**: Сигмоїдальне ядро корисно, коли дані мають нестандартні форми та взаємозв'язки

Як ми бачимо, що кожен із ядер має власні переваги й мінуси, саме в нашому випадку найкраще себе показав метод із Гаусовим ядром.

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("Перші п'ять рядків з масиву даних:")
print(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)
Перші п'ять рядків з масиву даних:
[[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 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]
```

Рис.5 Результат виконання програми

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
from sklearn.datasets import load_iris

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)
print(dataset.head(20))
print(dataset.describe())
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="accuracy")
    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))

# Нові дані для класифікації (за вашими значеннями)
X_new = np.array([[5.0, 2.9, 1.0, 0.2]])

# Зробіть прогноз для нових даних
prediction = model.predict(X_new)

# Виведіть результат прогнозу
print("Прогноз: {}".format(prediction))
```

Результат виконання програми:

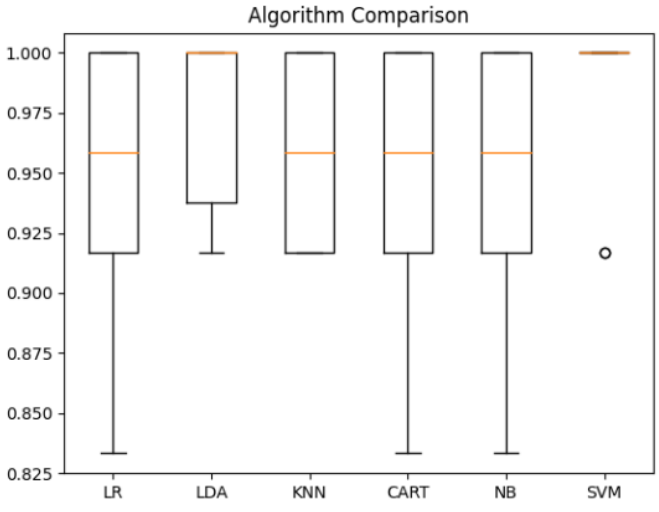

Рис.5 Візуалізація

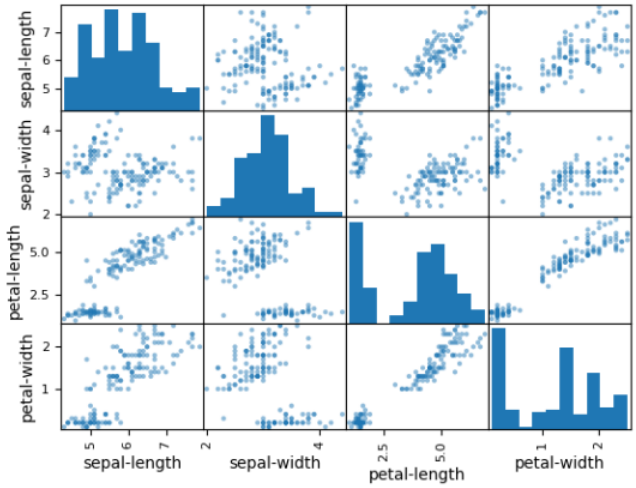

Рис.6 Візуалізація

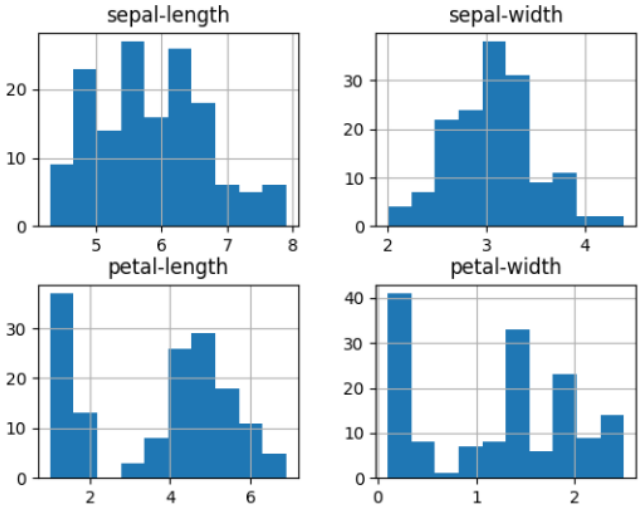

Рис.7 Візуалізація

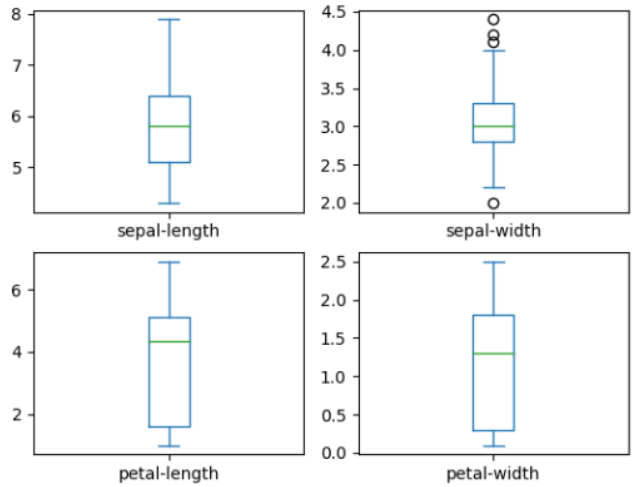

Рис.8 Візуалізація

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг програми:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn import preprocessing
import numpy as np


# 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)

# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```python
# Ініціалізація та навчання моделей
models = [
    ("LR", LogisticRegression()),
    ("LDA", LinearDiscriminantAnalysis()),
    ("KNN", KNeighborsClassifier()),
    ("CART", DecisionTreeClassifier()),
    ("NB", GaussianNB()),
    ("SVM", SVC())
]

results = []

for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

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

    results.append((name, accuracy, precision, recall, f1))

# Виведення результатів для порівняння
for name, accuracy, precision, recall, f1 in results:
    print(f"{name}:")
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-score: {f1:.2f}")
    print()
```

Результат виконання програми:

```
LR:
Accuracy: 0.79
Precision: 0.79
Recall: 0.79
F1-score: 0.76

LDA:
Accuracy: 0.81
Precision: 0.80
Recall: 0.81
F1-score: 0.79

KNN:
Accuracy: 0.76
Precision: 0.73
Recall: 0.76
F1-score: 0.74

CART:
Accuracy: 0.81
Precision: 0.81
Recall: 0.81
F1-score: 0.81
```

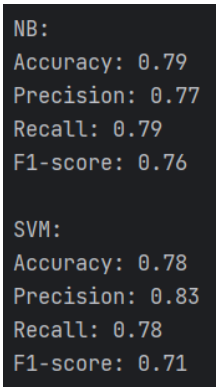

Рис.9 Результат виконання програми

Перед тим, як робити висновки, давайте трішки опишемо кожен із методів:

1. **Логістична регресія (LR)**: Цей метод використовує логістичну функцію для прогнозування ймовірності належності до класу. Він часто використовується для бінарної та багатокласової класифікації і добре працює для лінійно роздільних даних.
2. **Лінійний дискримінантний аналіз (LDA)**: LDA також працює з лінійно роздільними даними і намагається знайти лінійну комбінацію ознак, яка максимізує відстань між класами.
3. **Метод k-найближчих сусідів (KNN)**: KNN використовує найближчих сусідів об'єкта для класифікації. Він добре працює для даних зі складною структурою та нелинійними залежностями.
4. **Класифікація та регресія за допомогою дерев (CART)**: CART використовує рішальні дерева для розділення даних на різні класи. Вони можуть бути лінійними або нелинійними, залежно від структури дерева.
5. **Наївний баєсовський класифікатор (NB)**: Цей метод ґрунтується на теорії ймовірності і передбачає клас на основі наївного припущення про незалежність між ознаками.
6. **Метод опорних векторів (SVM)**: SVM намагається знайти гіперплощину, яка найкращим чином розділяє дані на класи. Він може використовуватися як для лінійно роздільних, так і для нелинійно роздільних даних за допомогою ядер.

Як ми бачимо, кожен з них використовується у своїх власних ситуаціях й працює різними методами. Саме в нашому випадку найкраще себе показав метод класифікації та регресії за допомогою дерев.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг програми:

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

iris = load_iris()
X, y = iris.data, iris.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3,
random_state=0)

clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)
ypred = clf.predict(Xtest)

accuracy = np.round(metrics.accuracy_score(ytest, ypred), 4)
precision = np.round(metrics.precision_score(ytest, ypred, average='weighted'), 4)
recall = np.round(metrics.recall_score(ytest, ypred, average='weighted'), 4)
f1_score = np.round(metrics.f1_score(ytest, ypred, average='weighted'), 4)
cohen_kappa = np.round(metrics.cohen_kappa_score(ytest, ypred), 4)
matthews_corrcoef = np.round(metrics.matthews_corrcoef(ytest, ypred), 4)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1_score)
print('Cohen Kappa Score:', cohen_kappa)
print('Matthews Corrcoef:', matthews_corrcoef)

classification_report = metrics.classification_report(ytest, ypred)
print('\t\tClassification Report:\n', classification_report)

mat = confusion_matrix(ytest, ypred)
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 програми:

```
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
```

Рис.10 Результат виконання програми

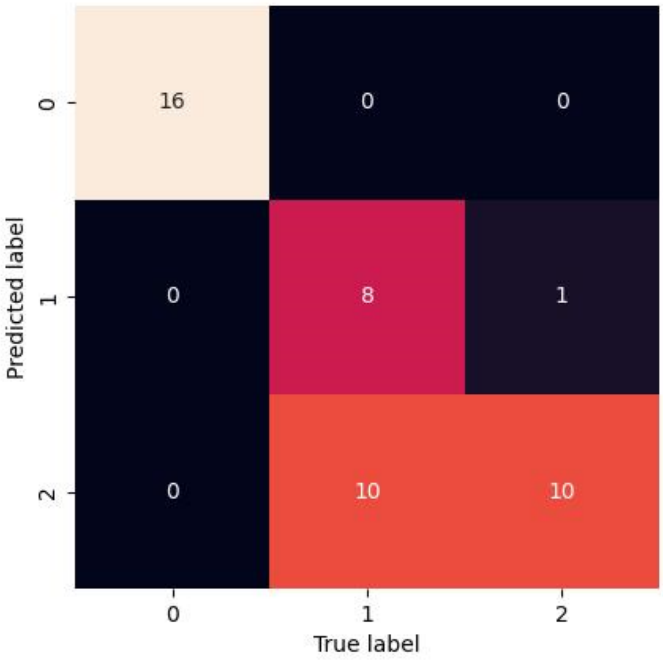

Рис.11 Візуалізація

Коефіцієнт Коена Каппа (Cohen's Kappa) і коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient) є метриками, які використовуються для оцінки якості класифікаційних моделей, особливо в задачах бінарної класифікації. Вони дозволяють враховувати не тільки правильні класифікації, але й помилкові класифікації та дисбаланс класів.

Висновок: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.

Посилання на GitHub: https://github.com/BOYYYKO/ai