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.

GitHub = https://github.com/Ozzornin/AIS/tree/master

- **age**: Вік – числова (ціла) ознака.
- **workclass**: Клас роботи (Income) – категоріальна ознака.
- **fnlwgt**: Вага вибірки (final weight) – числова (ціла) ознака.
- **education**: Освітній рівень – категоріальна ознака.
- **education-num**: Рівень освіти (у числовому вигляді) – числова (ціла) ознака.
- **marital-status**: Сімейний статус – категоріальна ознака.
- **occupation**: Професія – категоріальна ознака.
- **relationship**: Відношення в сім'ї – категоріальна ознака.
- **race**: Раса – категоріальна ознака.
- **sex**: Стать – бінарна ознака.
- **capital-gain**: Прибуток від капіталу – числова (ціла) ознака.
- **capital-loss**: Втрата від капіталу – числова (ціла) ознака.
- **hours-per-week**: Кількість робочих годин на тиждень – числова (ціла) ознака.
- **native-country**: Рідна країна – категоріальна ознака.

LR_2_task_1.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import LinearSVC

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)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))

classifier.fit(X, y)

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))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

cv = 3
accuracy_values = cross_val_score(classifier, X, y, scoring="accuracy", cv=cv)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, y, scoring="precision_weighted", cv=cv
)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring="recall_weighted",
cv=cv)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1 = cross_val_score(classifier, X, y, scoring="f1_weighted", cv=3)
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

input_data = [
    "37",
    "Private",
    "215646",
    "HS-grad",
    "9",
    "Never-married",
    "Handlers-cleaners",
    "Not-in-family",
    "White",
```

```
    "Male",
    "0",
    "0",
    "40",
    "United-States",
]

input_data_encoded = [-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)

predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
PS D:\Study\zdtu\AIS\lab2> & C:/Users/yozor/AppData/Local/Programs/Python/Python312/python.exe d:/Study/zdtu/AIS/lab2/LR_2_task_1.py
Accuracy: 79.66%
Precision: 78.88%
Recall: 79.66%
F1 score: 76.01%
<=50K
```

Рис 1 – показники якості

Точка відноситься до класу <=50k

LR_2_task_2.1.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC

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)


classifier = OneVsOneClassifier(SVC(kernel="poly", degree=8))

classifier.fit(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(SVC(kernel="poly", degree=8))
print("before fit")
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("after fit")
cv = 3
accuracy_values = cross_val_score(classifier, X, y, scoring="accuracy", cv=cv)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, y, scoring="precision_weighted", cv=cv
)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring="recall_weighted",
cv=cv)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1 = cross_val_score(classifier, X, y, scoring="f1_weighted", cv=3)
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

input_data = [
    "37",
    "Private",
    "215646",
```

```
    "HS-grad",
    "9",
    "Never-married",
    "Handlers-cleaners",
    "Not-in-family",
    "White",
    "Male",
    "0",
    "0",
    "40",
    "United-States",
]

input_data_encoded = [-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)

predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
PS D:\Study\zdtu\AIS\lab2> & C:/Users/yozor/AppData/Local/Programs/Python/Python312/python.exe d:/Study/zdtu/AIS/lab2/LR_2_task_2_1.py
before fit
after fit
Accuracy: 61.68%
Precision: 74.13%
Recall: 61.68%
F1 score: 51.48%
<=50K
```

Рис 2 – Оцінка якості моделі з поліоміальним ядром

LR_2_task_2.2.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC

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)


classifier = OneVsOneClassifier(SVC(kernel="rbf"))

classifier.fit(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(SVC(kernel="rbf"))
print("before fit")
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("after fit")
cv = 3
accuracy_values = cross_val_score(classifier, X, y, scoring="accuracy", cv=cv)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, y, scoring="precision_weighted", cv=cv
)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring="recall_weighted",
cv=cv)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1 = cross_val_score(classifier, X, y, scoring="f1_weighted", cv=3)
```

```
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

input_data = [
    "37",
    "Private",
    "215646",
    "HS-grad",
    "9",
    "Never-married",
    "Handlers-cleaners",
    "Not-in-family",
    "White",
    "Male",
    "0",
    "0",
    "40",
    "United-States",
]

input_data_encoded = [-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)

predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
PS D:\Study\zdtu\AIS\lab2> & C:/Users/yozor/AppData/Local/Programs/Python/Python312/python.exe d:/Study/zdtu/AIS/lab2/LR_2_task_2_2.py
before fit
after fit
Accuracy: 78.61%
Precision: 83.06%
Recall: 78.61%
F1 score: 71.95%
<=50K
```

Рис 3 – Оцінка якості моделі з гаусівським ядром

LR_2_task_2.3.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC

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)


classifier = OneVsOneClassifier(SVC(kernel="sigmoid"))

classifier.fit(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(SVC(kernel="sigmoid"))
print("before fit")
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("after fit")
cv = 3
accuracy_values = cross_val_score(classifier, X, y, scoring="accuracy", cv=cv)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, y, scoring="precision_weighted", cv=cv
```

```
)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring="recall_weighted",
cv=cv)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1 = cross_val_score(classifier, X, y, scoring="f1_weighted", cv=3)
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

input_data = [
    "37",
    "Private",
    "215646",
    "HS-grad",
    "9",
    "Never-married",
    "Handlers-cleaners",
    "Not-in-family",
    "White",
    "Male",
    "0",
    "0",
    "40",
    "United-States",
]

input_data_encoded = [-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)

predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
PS D:\Study\zdtu\AIS\lab2> & C:/Users/yozor/AppData/Local/Programs/Python/Python312/python.exe d:/Study/zdtu/AIS/lab2/LR_2_task_2_3.py
before fit
after fit
Accuracy: 63.89%
Precision: 63.65%
Recall: 63.89%
F1 score: 63.77%
<=50K
```

Рис. 4 – Оцінка якості ядра з сигмоїдальним ядром

Полягаючись на оцінки якості, із цих трьох моделей з різними ядрами найкраще впоралася модель з гаусівським ядром.

LR_2_task_3.1.py

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

print("Ключі iris_dataset: \n{}\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("Тип масиву target:{}".format(type(iris_dataset["target"])))
print("Відповіді:\n{}\n".format(iris_dataset["target"]))

print(format(iris_dataset["feature_names"]))
for i in range(5):
    print("{}".format(i + 1) + ") " + "{}".format(iris_dataset["data"][i]))
```

LR_2_task_3.2.py

Я обрав SVM, тому що це універсальний і потужний алгоритм, який може моделювати як лінійні, так і нелінійні зв'язки в даних.

Також SVM має найкращі показники серед інших алгоритмів.

Квітка з кроку 8 належить до класу `Iris-setosa`

```
import matplotlib
import matplotlib.pyplot as pyplot
import numpy as np
from pandas import read_csv
from pandas.plotting import scatter_matrix
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.model_selection import StratifiedKFold, cross_val_score,
train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

matplotlib.use("Agg")


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.savefig("fig1.png")
pyplot.clf()


dataset.hist()
pyplot.savefig("fig2.png")
pyplot.clf()


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
)


scatter_matrix(dataset)
pyplot.savefig("fig3.png")
pyplot.clf()


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


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


pyplot.boxplot(results, tick_labels=names)
pyplot.title("Algorithm Comparison")
pyplot.savefig("fig4.png")
pyplot.clf()
```

```
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 X_new:", X_new.shape)
prediction = model.predict(X_new)
print("Прогноз: {}".format(prediction))
predicted_class = prediction[0]
print("Мітка: {}".format(predicted_class))
```

```
(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: 0.941667 (0.038188)
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 X_new: (1, 4)
Прогноз: ['Iris-setosa']
Мітка: Iris-setosa
```

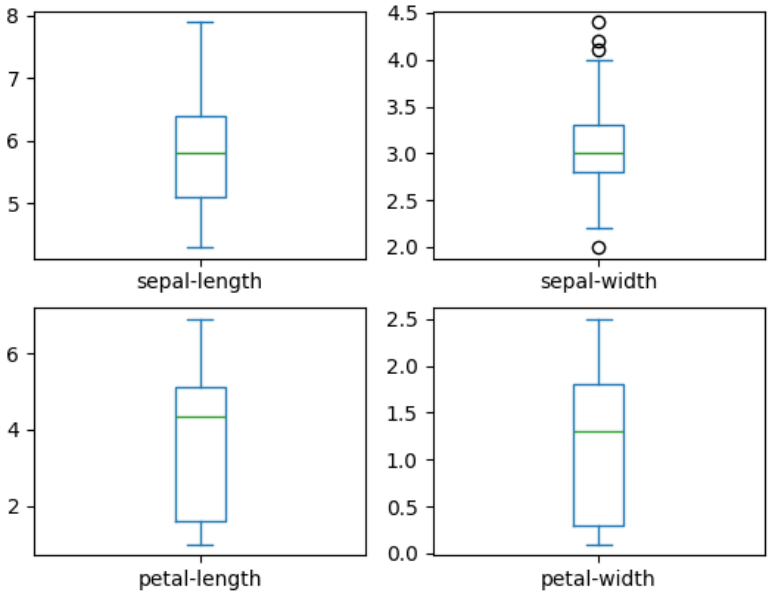


Рис. 5 – розподіл атрибутів на старті

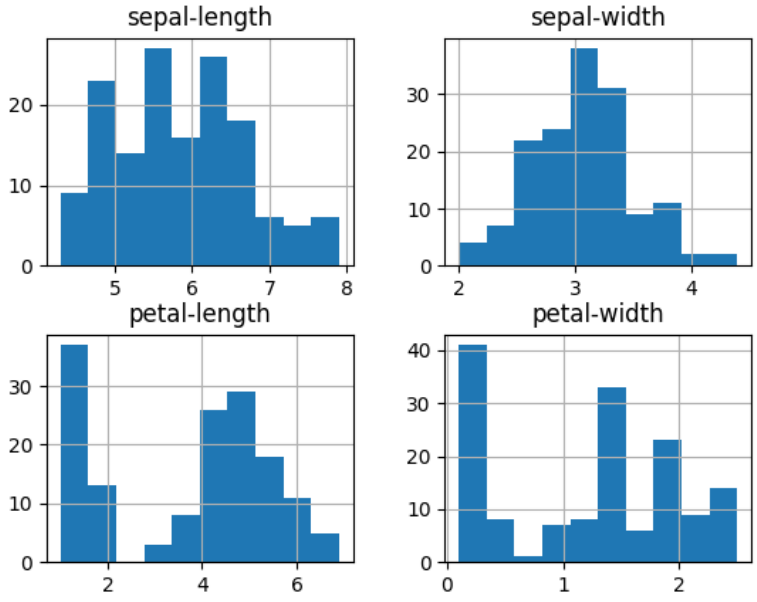


Рис. 6 – діаграма розподілу атрибутів

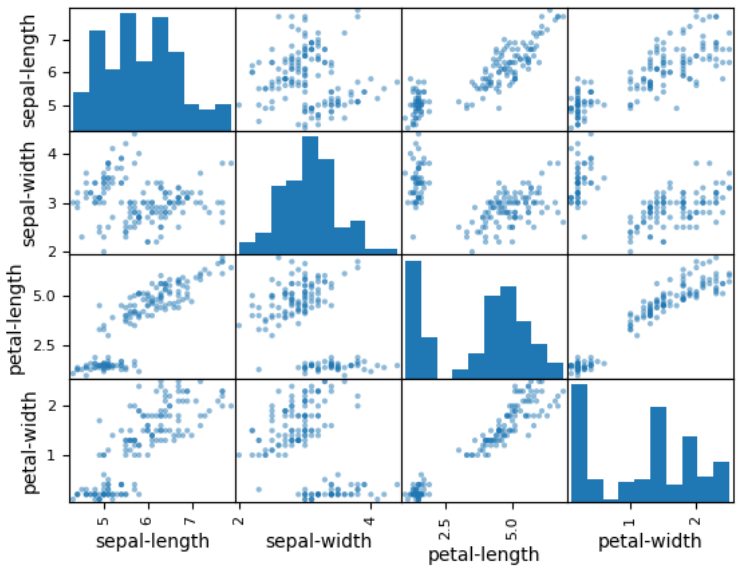


Рис. 7 – матриця діаграм розсіювання

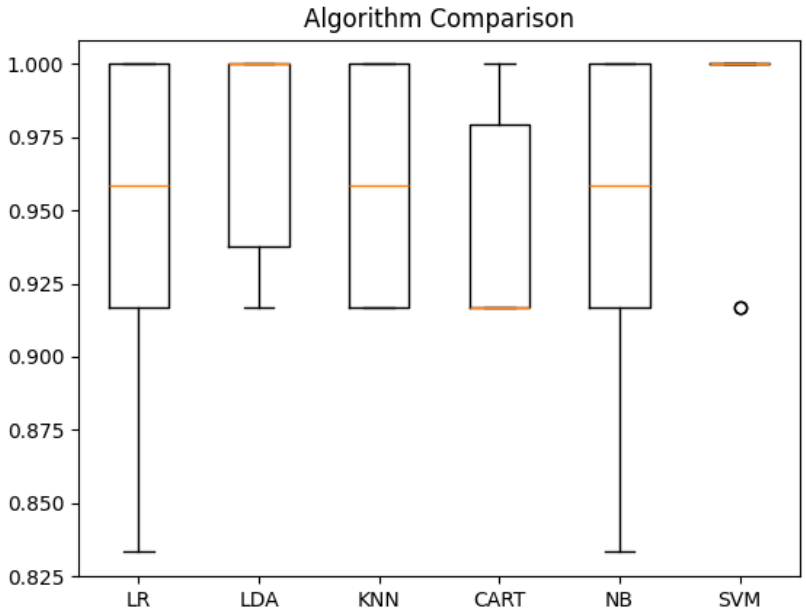


Рис. 8 – результати кожного алгоритму

LR_2_task_4.py

```
import pandas as pd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold, cross_val_score,
train_test_split
from sklearn.multiclass import OneVsRestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

data_path = "income_data.txt"
column_names = [
    "Age",
    "Workclass",
    "fnlwgt",
    "Education",
    "Education_Num",
    "Marital_Status",
    "Occupation",
    "Relationship",
    "Race",
    "Sex",
    "Capital_Gain",
    "Capital_Loss",
    "Hours_Per_Week",
    "Native_Country",
    "Income",
```

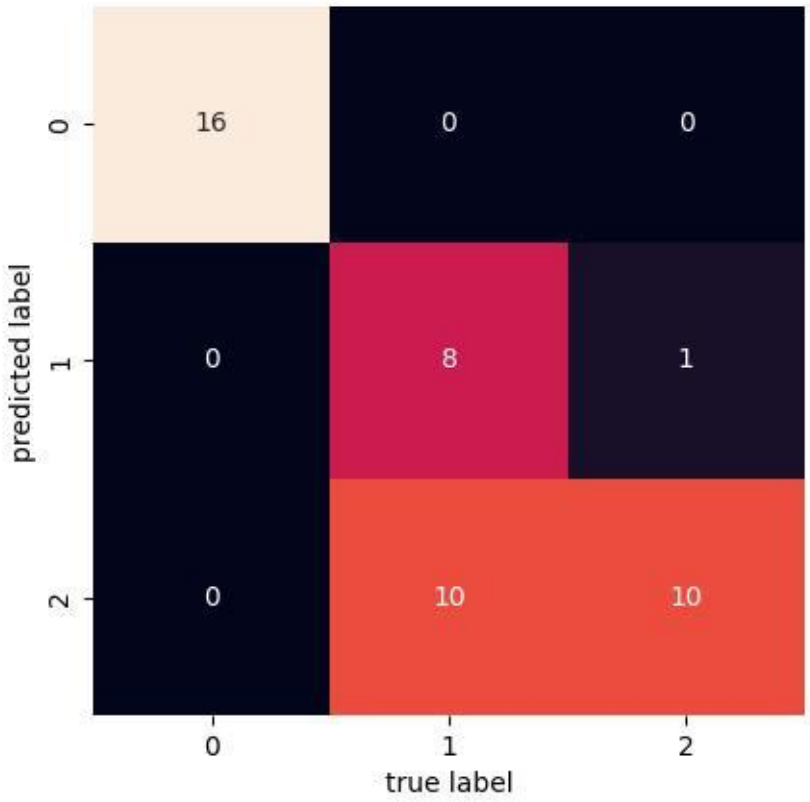
```
]
data = pd.read_csv(data_path, sep=",", header=None, names=column_names)

categorical_columns = [
    "Workclass",
    "Education",
    "Marital_Status",
    "Occupation",
    "Relationship",
    "Race",
    "Sex",
    "Native_Country",
    "Income",
]

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
for column in categorical_columns:
    data[column] = encoder.fit_transform(data[column])

features = data.drop("Income", axis=1)
target = data["Income"]

X_train, X_test, y_train, y_test = train_test_split(
    features, target, test_size=0.3, random_state=42
)

classifiers = {
    "Logistic Regression":
OneVsRestClassifier(LogisticRegression(solver="liblinear")),
    "Linear Discriminant": LinearDiscriminantAnalysis(),
    "K-Neighbors Classifier": KNeighborsClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "Gaussian Naive Bayes": GaussianNB(),
    "Support Vector Machine": SVC(gamma="scale"),
}

for classifier_name, model in classifiers.items():
    kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
    scores = cross_val_score(model, X_train, y_train, cv=kfold,
scoring="accuracy")
    print(
        f"{classifier_name}: Mean Accuracy = {scores.mean():.4f}, Std Dev =
{scores.std():.4f}"
    )
```

```
Logistic Regression: Mean Accuracy = 0.7911, Std Dev = 0.0079
Linear Discriminant: Mean Accuracy = 0.8140, Std Dev = 0.0038
K-Neighbors Classifier: Mean Accuracy = 0.7707, Std Dev = 0.0045
Decision Tree: Mean Accuracy = 0.8133, Std Dev = 0.0061
Gaussian Naive Bayes: Mean Accuracy = 0.7923, Std Dev = 0.0049
Support Vector Machine: Mean Accuracy = 0.7921, Std Dev = 0.0036
```

```
from io import BytesIO

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

matplotlib.use("Agg")

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)

ypred = clf.predict(X_test)

print("Accuracy:", np.round(metrics.accuracy_score(y_test, ypred), 4))
print(
    "Precision:",
    np.round(metrics.precision_score(y_test, ypred, average="weighted"), 4),
)
print("Recall:", np.round(metrics.recall_score(y_test, ypred,
average="weighted"), 4))
print("F1 Score:", np.round(metrics.f1_score(y_test, ypred, average="weighted"),
4))
print("Cohen Kappa Score:", np.round(metrics.cohen_kappa_score(y_test, ypred),
4))
print("Matthews Corrcoef:", np.round(metrics.matthews_corrcoef(y_test, ypred),
4))
print("\t\tClassification Report:\n", metrics.classification_report(ypred,
y_test))


mat = confusion_matrix(y_test, 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")

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

```
plt.savefig("Confusion.svg")
```

```
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
```



Матриця плутанини показує, як класифікатор передбачив класи порівняно з реальними значеннями.

На зображенні матриця 3x3 для трьох класів (0, 1, 2):

Горизонтально - істинні класи.

Вертикально - передбачені класи.

Правильні передбачення:

Клас 0 -16 з 16.

Клас 1 - 8 з 9, 1 помилково як клас 2.

Клас 2 - 10 правильних, 9 помилково як клас 1.

Налаштування класифікатора:

tol=1e-2: Це параметр допустимої похибки, він визначає, наскільки зміна ваг між ітераціями може бути малою, щоб вважати модель сконвергованою

solver="sag": Це метод для розв'язку оптимізаційної задачі. sag

Коефіцієнт кореляції Метьюза - Показник якості бінарної класифікації, що враховує як правильні передбачення, так і помилки для всіх класів, і більш точний для оцінки роботи моделі з неврівноваженими класами.

Коефіцієнт Коена Каппа - Вимірює рівень узгодженості між реальними та передбаченими класами, враховуючи випадкові збіги. Значення від -1 (повна невідповідність) до 1 (повна відповідність).

ВИСНОВОК

У рамках лабораторної роботи було досліджено різні методи класифікації даних. Проведений аналіз показав, що кожен метод має свої переваги та недоліки, залежно від характеристик даних і задачі.