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 1.

1. Age - (числова) вік.
2. Workclass - (категоріальна) робочий клас: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. Fnlwgt - (числова) final weight, кількість людей, яку представляє запис.
4. Education - (категоріальна) Рівень освіти: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5. Education-num - (числова) кількість освіт.
6. marital-status - (категоріальна) сімейний стан: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. occupation - (категоріальна) Професія: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
8. relationship - (категоріальна) Роль у сім'ї: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. race - (категоріальна) – Раса: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. sex – (бінарна) Стать.
11. capital-gain - (числова) Дохід.
12. capital-loss - (числова) Витрати.
13. hours-per-week - (числова) К-ть робочих годин на тиждень.
14. native-country - (категоріальна) Рідна країна.
15. income – (бінарна) Дохід >50K або <=50K.

Лістинг LR_2_task_1.py:

```
import numpy as np
import matplotlib.pyplot as plt
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



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

        if income_class == '>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:
        current_label_encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)
```

```
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
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)) + "%")

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:
        encoder = label_encoder[count]
        input_data_encoded[i] = int(encoder.transform([(input_data[i])])[-1])
        count += 1

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

Результат виконання програми:

```
Accuracy score: 62.64%
Precision score: 69.18%
Recall score: 38.24%
F1 score: 56.15%
<=50K
```

Висновок роботи програми: тестова точка належить до класу <=50K

<=50K

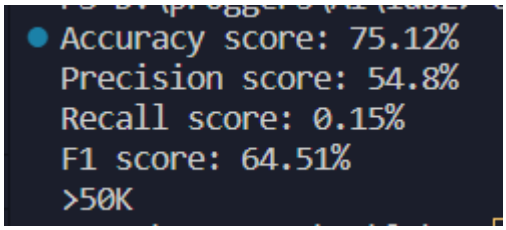
Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Лістинг програми LR_2_task_2_1.py (Поліноміальне):

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_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(', ')
        income_class = data[-1]
        if income_class == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

        if income_class == '>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:
        current_label_encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)
```

```
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, max_iter=10000))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
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)) + "%")

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:
        encoder = label_encoder[count]
        input_data_encoded[i] = int(encoder.transform([(input_data[i])])[-1])
        count += 1

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

Результат виконання poly:



Лістинг програми LR_2_task_2_2.py (Гаусове):

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_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(', ')
        income_class = data[-1]
        if income_class == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

        if income_class == '>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:
        current_label_encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(SVC(kernel='rbf', max_iter=10000))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
classifier.fit(X_train, y_train)
```

```
y_test_pred = classifier.predict(X_test)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
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)) + "%")

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:
        encoder = label_encoder[count]
        input_data_encoded[i] = int(encoder.transform([(input_data[i])])[-1])
        count += 1

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

Результат виконання rbf:

```
PS D:\proggers\AI\lab2>
Accuracy score: 78.61%
Precision score: 98.72%
Recall score: 14.26%
F1 score: 71.95%
<=50K
```

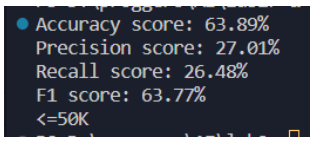
Лістинг програми LR_2_task_2_3.py (Сигмоїдальне):

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_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(', ')
        income_class = data[-1]
        if income_class == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

        if income_class == '>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:
        current_label_encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(SVC(kernel='sigmoid', max_iter=10000))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
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)) + "%")

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:
        encoder = label_encoder[count]
        input_data_encoded[i] = int(encoder.transform([(input_data[i])])[-1])
        count += 1

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

Результат виконання sigmoid:



Для усіх трьох видів SVM було задано пеараметр max-iter=10000.

Висновок: Гаусове (RBF) ядро має найвищий показник точності (Accuracy) серед усіх ядер і найвищий показник точності класифікації. Він також має високий показник Precision, що означає, що модель добре виділяє позитивні класи. Однак, Recall у цього ядра не є дуже високим, що може означати, що воно може пропускати деякі позитивні приклади. F1-score також вищий, ніж у лінійного ядра.

Поліноміальне ядро має високий показник Accuracy, але відмінності в Precision та Recall є дуже великими, що може свідчити про дуже нерівномірну класифікацію. Висока точність може бути обумовлена великою кількістю правильно класифікованих негативних прикладів.

Сигмоїдальне ядро має низький показник точності (Accuracy) та Precision, що може означати, що це ядро слабко підходить для даного завдання класифікації.

З усього цього можна зробити висновок, що гаусове (RBF) ядро має найкращі результати.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Лістинг коду ознайомлення зі структурою даних:

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

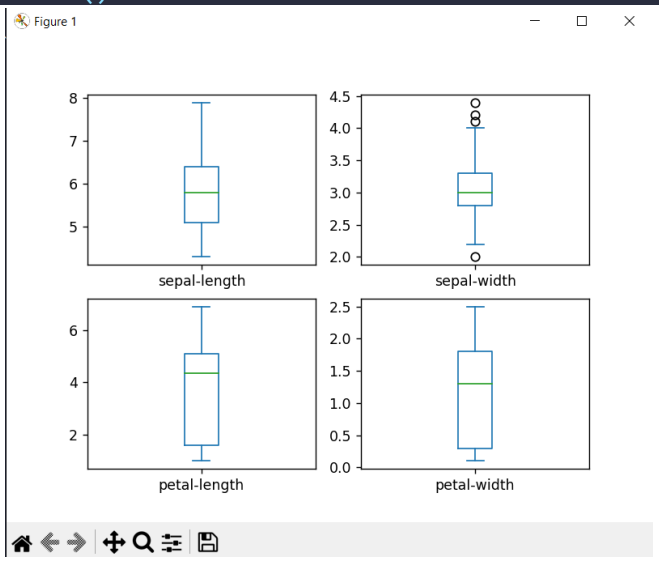
Результат:

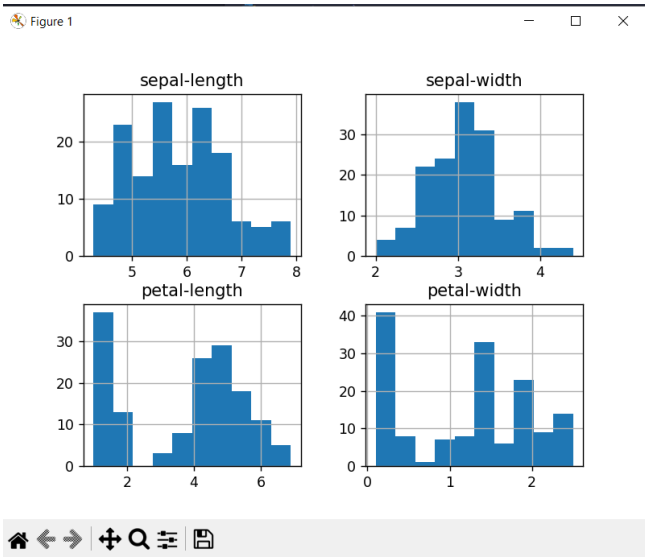
```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

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)
[[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 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]
```

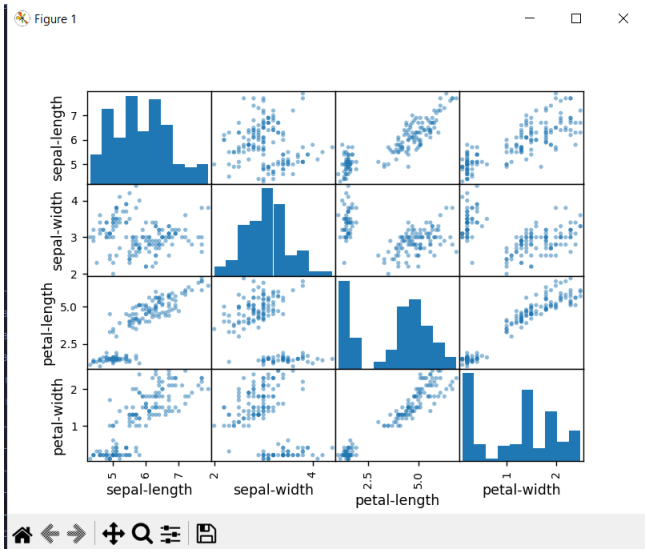
Код візуалізації графіків:

```
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)

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()
```





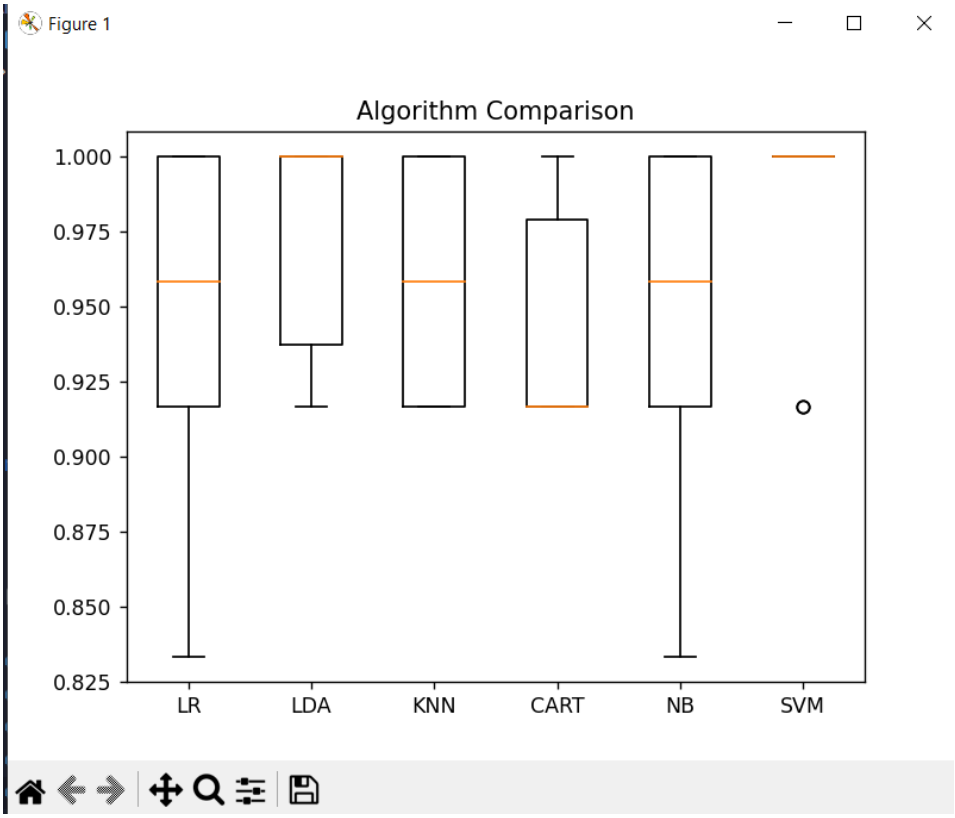


Лістинг програми:

```
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


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()

# step 3
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=0.2,
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='accura-
cy')
    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()
```

Результат:




```
(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)
```

SVM має найвищий показник точності (Accuracy) - 98.33%. Таким чином, на основі цих результатів можна вважати, що метод опорних векторів (SVM) найкраще впорався з завданням класифікації набору даних.аними.

Лістинг програми LR_2_task_3.py:

```
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()

# step 3
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=0.2,
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='accura-
cy')
    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))

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("Спрогнозированная метка: {}".format(prediction[0]))
```

Результа виконання програми:

```
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']
Спрогнозированная метка: Iris-setosa
```

Висновок: Оцінка точності (accuracy) дял обраного методу SVM – 96.67%. Квітка з кроку 8 ([5, 2.9, 1, 0.2]) належить до класу Iris-setosa.

Завдання 2.4 Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг програми LR_2_task_4.py:

```python
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
```

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

        if income_class == '>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:
        current_label_encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.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 = []
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: %s (%f)' % (name,str(round(100 * cv_results.mean(), 2)) + "%",cv_re-
sults.std()))
```

Результат виконання:

```
LR: 79.34% (0.006253)
LDA: 81.22% (0.003802)
KNN: 76.7% (0.006871)
CART: 80.6% (0.006870)
NB: 78.98% (0.004791)
SVM: 75.38% (0.001207)
```

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг програми LR_2_task_5.py:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from io import BytesIO
import seaborn as sns
import matplotlib.pyplot as plt
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)
print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))
print('Precision:', np.round(metrics.precision_score(ytest, ypred,aver-
age='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(ytest, ypred, average='weighted'),4))
print('F1 Score:', np.round(metrics.f1_score(ytest, ypred, average='weighted'),4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(ytest, ypred), 4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n', metrics.classification_report(ypred, ytest))
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")
```

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

Результат виконання:

```
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
```

У класифікаторі Ridge були використані налаштування tol=1e-2 – точність, solver="sag" – розв'язник Stochastic Average Gradient.

Показники якості:

Акуратність – 75.56%

Точність – 83.33%

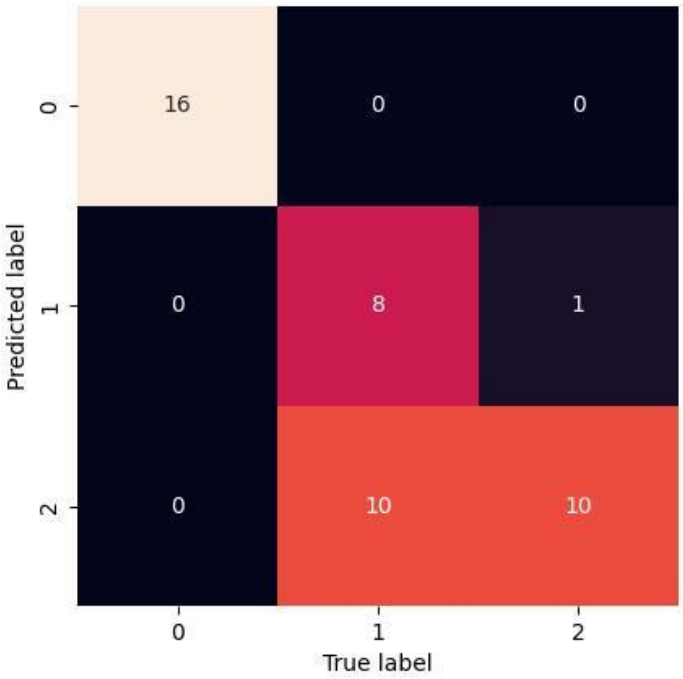
Повнота – 75.56%

f-міра – 75.03%

коефіцієнт Коена Каппа – 64.31%

коефіцієнт кореляції Метьюза – 68.31%

Confusion.jpg:



Це матриця плутанини (confusion matrix) – візуалізація результату класифікації моделі машинного навчання. На осі x відображаються значення "True label" (правильних міток), а на осі y відображаються значення "Predicted label" (прогнозованих міток)

Коефіцієнт Коена Каппа вимірює ступінь узгодженості між прогнозованими класифікаціями і дійсними класами у випадку, коли існує випадкова вірогідність вибору класу.

Коефіцієнт кореляції Метьюза вимірює якість класифікаційної моделі, якщо дані мають баланс або дисбаланс класів.

Github: ***https://github.com/mtvi/ipz202_Shcherback_lab2***

Висновки: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчилися їх порівнювати.