

**Міністерство освіти та науки України  
Київський національний економічний університет  
імені Вадима Гетьмана**

**Кафедра інформаційних систем  
в економіці**

**Дисципліна “ Системи і методи штучного інтелекту”**

**ЗВІТ  
з лабораторної роботи №2**

**Підготував:**

студент курсу

групи ІН\_402

спеціальності “Комп’ютерні науки”

ПІБ студента	Тест	Звіт	Захист	Сума балів
Сіденко Д.С.				

**Прийняв:**

Волошин А.П.

**Київ -2025**

Тема: Порівняння методів класифікації даних

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

Посилання на гітхаб: <https://github.com/dimon2202/Artificial-intelligence-systems-and-methods.git>

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

Набір даних в цьому завданні має такі ознаки:

1. **Age:** Вік особи (числовий, цілочисловий).
2. **Workclass:** Тип зайнятості (категоріальний).
3. **Fnlwgt:** Вага зразка — специфічний фактор ваги в популяційному аналізі (числовий, цілочисловий).
4. **Education:** Рівень освіти (категоріальний).
5. **Education-num:** Кількість років освіти (числовий, цілочисловий).
6. **Marital-status:** Сімейний статус (категоріальний).
7. **Occupation:** Рід діяльності (категоріальний).
8. **Relationship:** Сімейна роль (категоріальний).
9. **Race:** Раса особи (категоріальний).
10. **Sex:** Стать (категоріальний).
11. **Capital-gain:** Прибуток від капіталовкладень (числовий, цілочисловий).
12. **Capital-loss:** Збиток від капіталовкладень (числовий, цілочисловий).
13. **Hours-per-week:** Кількість робочих годин на тиждень (числовий, цілочисловий).
14. **Native-country:** Рідна країна (категоріальний).
15. **Income:** Клас доходу (категоріальний, дві категорії: " $\leq 50K$ " або " $> 50K$ ").

Результат виконання програми:

```
Accuracy: 79.56%  
Precision: 79.26%  
Recall: 79.56%  
F1 Score: 75.75%  
Predicted class: <=50K
```

Програмний код:

```
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.metrics._classification import accuracy_score, precision_score,  
recall_score, f1_score  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model_selection import train_test_split, cross_val_score  
  
# Вхідний файл, який містить дані  
input_file = 'income_data.txt'  
  
# Читання даних  
X = []  
Y = []  
count_class1 = 0  
count_class2 = 0  
max_datapoints = 25000  
  
with open(input_file, 'r') as f:  
    for line in f.readlines():  
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:  
            break  
        if '?' in line:  
            continue  
        data = line.strip().split(' ', '  
        if data[-1] == '<=50K' and count_class1 < max_datapoints:  
            X.append(data[:-1])  
            Y.append('<=50K')  
            count_class1 += 1  
        elif data[-1] == '>50K' and count_class2 < max_datapoints:  
            X.append(data[:-1])  
            Y.append('>50K')  
            count_class2 += 1  
  
# Перетворення на масив numpy  
X = np.array(X)  
Y = np.array(Y)
```

```

# Перетворення рядкових даних на числові
label_encoders = []
X_encoded = np.empty(X.shape)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders.append(le)

# Поділ на навчальні і тестові дані
X_train, X_test, y_train, y_test = train_test_split(X_encoded, Y, test_size=0.2,
random_state=5)

# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))

# Навчання класифікатора
classifier.fit(X_train, y_train)

# Передбачення для тестових даних
y_test_pred = classifier.predict(X_test)

# Обчислення F-міри для SVM-класифікатора
accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted')
recall = recall_score(y_test, y_test_pred, average='weighted')
f1 = f1_score(y_test, y_test_pred, average='weighted')

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

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-
States']

# Кодування тестової точки даних
input_data_encoded = np.array([-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] = label_encoders[count].transform([item])[0]
        count += 1

```

```
input_data_encoded = input_data_encoded.reshape(1, -1)

# Використання класифікатора для передбачення точки даних
predicted_class = classifier.predict(input_data_encoded)
print("Predicted class:", predicted_class[0])
```

Судячи по результату, тестова точка належить до групи що отримує менше ніж 50 тисяч в рік...

## Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Програмний код поліноміального ядра:

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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.strip().split(',')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data[:-1])
            Y.append('<=50K')
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data[:-1])
            Y.append('>50K')
            count_class2 += 1
```

```

# Перетворення на масив numpy
X = np.array(X)
Y = np.array(Y)

# Перетворення рядкових даних на числові
label_encoders = []
X_encoded = np.empty(X.shape)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders.append(le)

# Поділ на навчальні і тестові дані
X_train, X_test, y_train, y_test = train_test_split(X_encoded, Y, test_size=0.2,
random_state=5)

# Функція для оцінки та виведення метрик
def evaluate_classifier(classifier, X_train, X_test, y_train, y_test):
    classifier.fit(X_train, y_train)
    y_test_pred = classifier.predict(X_test)

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

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

# Поліноміальне ядро
# Якщо я ставлю degree >2, воно тренує його 10 років. В degree 1 даже параметри
кращі ніж в 2
print("Поліноміальне ядро:")
classifier_poly = SVC(kernel='poly', degree=2)
evaluate_classifier(classifier_poly, X_train, X_test, y_train, y_test)

```

Результат:

```

Поліноміальне ядро:
Accuracy: 77.39%
Precision: 81.11%
Recall: 77.39%
F1 Score: 78.18%

```

## Програмний код гаусового ядра:

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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.strip().split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data[:-1])
            Y.append('<=50K')
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data[:-1])
            Y.append('>50K')
            count_class2 += 1

# Перетворення на масив numpy
X = np.array(X)
Y = np.array(Y)

# Перетворення рядкових даних на числові
label_encoders = []
X_encoded = np.empty(X.shape)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders.append(le)
```

```

# Поділ на навчальні і тестові дані
X_train, X_test, y_train, y_test = train_test_split(X_encoded, Y, test_size=0.2,
random_state=5)

# Функція для оцінки та виведення метрик
def evaluate_classifier(classifier, X_train, X_test, y_train, y_test):
    classifier.fit(X_train, y_train)
    y_test_pred = classifier.predict(X_test)

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

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

# Гаусове (RBF) ядро
print("Гаусове (RBF) ядро:")
classifier_rbf = SVC(kernel='rbf')
evaluate_classifier(classifier_rbf, X_train, X_test, y_train, y_test)

```

Результат:

```

Гаусове (RBF) ядро:
Акурау: 78.19%
Precision: 82.82%
Recall: 78.19%
F1 Score: 71.51%

```

Програмний код сигмоїдального ядра:

```

import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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.strip().split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data[:-1])
            Y.append('<=50K')
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data[:-1])
            Y.append('>50K')
            count_class2 += 1

# Перетворення на масив numpy
X = np.array(X)
Y = np.array(Y)

# Перетворення рядкових даних на числові
label_encoders = []
X_encoded = np.empty(X.shape)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders.append(le)

# Поділ на навчальні і тестові дані
X_train, X_test, y_train, y_test = train_test_split(X_encoded, Y, test_size=0.2,
random_state=5)

# Функція для оцінки та виведення метрик
def evaluate_classifier(classifier, X_train, X_test, y_train, y_test):
    classifier.fit(X_train, y_train)
    y_test_pred = classifier.predict(X_test)

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

    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Precision: {precision * 100:.2f}%")
    print(f"Recall: {recall * 100:.2f}%")

```

```

    print(f"F1 Score: {f1 * 100:.2f}%")
    print()
# Сигмоїдальне ядро
print("Сигмоїдальне ядро:")
classifier_sigmoid = SVC(kernel='sigmoid')
evaluate_classifier(classifier_sigmoid, X_train, X_test, y_train, y_test)

```

Результат:

```

Сигмоїдальне ядро:
Акурасу: 60.47%
Precision: 60.64%
Recall: 60.47%
F1 Score: 60.55%

```

По результатам видно що Поліномінальне та Гаусове ядра виконують поставлену задачу майже на схожому рівні, з перебісом в +- 1% по кожній метриці для Гаусового ядра. Сигмоїдальне впоралось набагато гірше. Краще всього буде вибрати гаусове ядро.

### Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Код для ознайомлення з датасетом:

```

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

```

Результат:

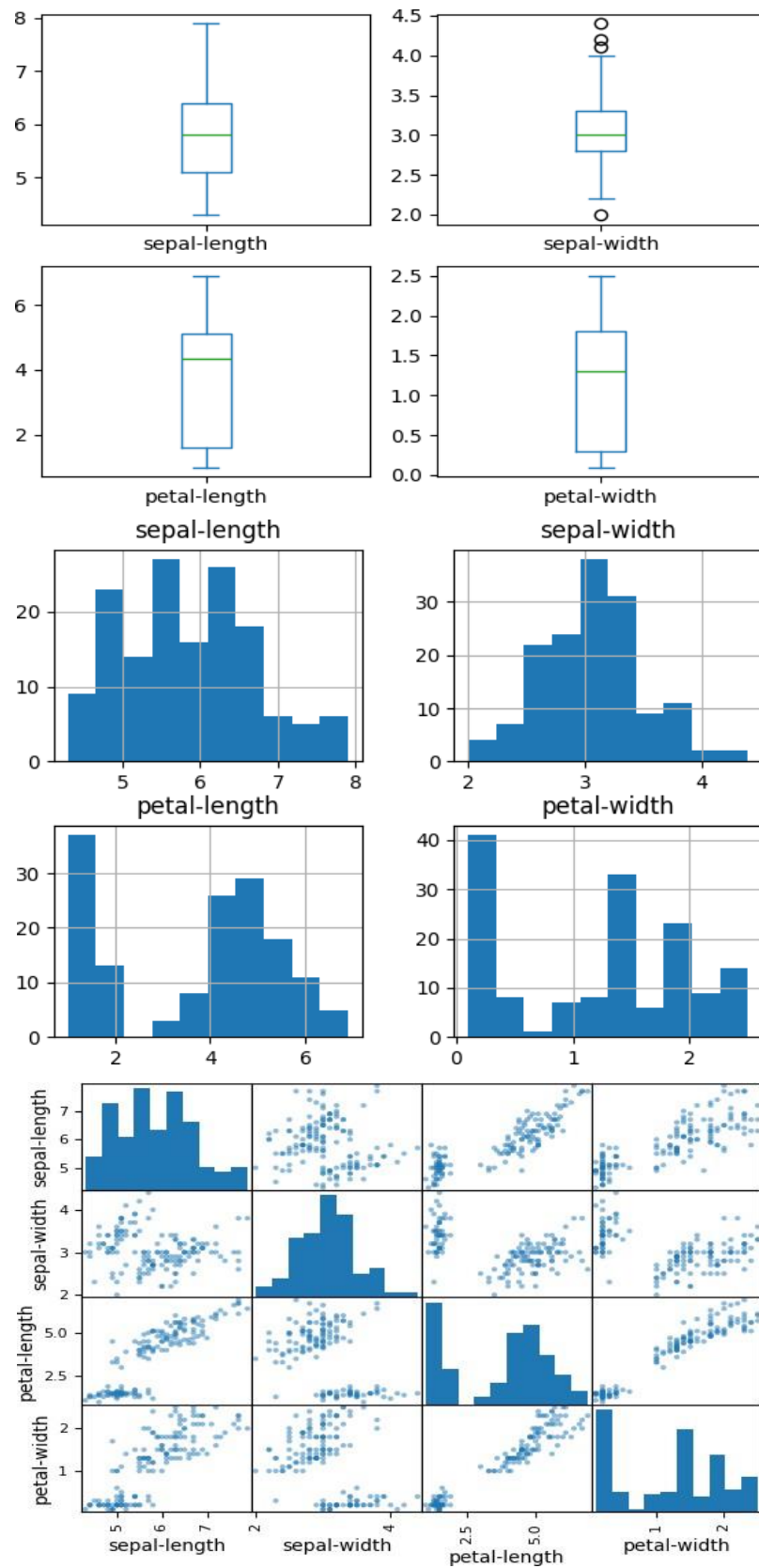
[illegible]

### Код для візуалізації:

```
# 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()
```



Візуалізовано графіки

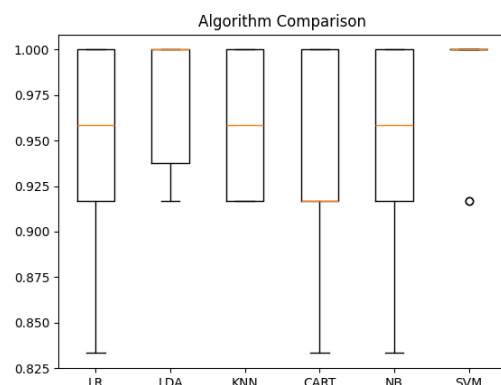
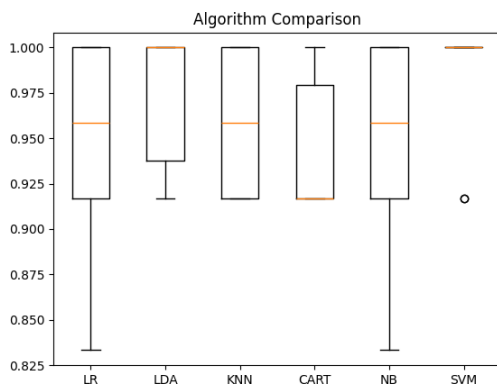
Код для перевірки точності алгоритмів:

```
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()))
```

Код для візуалізації порівняння алгоритмів:

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



Результати візуалізації

Можна побачити, що CART метод веде себе непередбачувано та постійно міняється після кожного запуску програми. Серед цих методів найбільш виділяється SVM, в ньому відсутній розподіл значень точності. Всі лінії знаходяться на рівні 1.0, що є ідеальним. Проте в ньому є один аутлаєр, який з'являється також стабільно. Думаю що вибрати SVM буде найкращим вибором, outlier можна проігнорувати в даному випадку.

Також алгоритми показали таку загальну точність:

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)

Отже, SVM показує себе найкраще.

```
# Прогнозування окремого входження за допомогою натренованої моделі
newdataset = np.array([[5, 2.9, 1, 0.2]])

svm_model = [model for name, model in models if name == 'SVM'][0]
svm_model.fit(X_train, Y_train)
prediction = svm_model.predict(newdataset)
print("Прогноз: " + str(prediction[0]))
```

Результат прогнозу:

Прогноз: Iris-setosa

Код програми:

```
# Завантаження бібліотек
from fontTools.misc.textTools import tostr
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.multiclass import OneVsRestClassifier
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)

# Зріз даних head
print(dataset.head(20))
# Статистичні зведення методом describe
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]
# Разделение X и y на обучающую и контрольную выборки
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(('LR', OneVsRestClassifier(LogisticRegression()))) #LR multi_class
був деприкейтед говорить
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()))

# Порівняння алгоритмів
pyplot.boxplot(results, tick_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("Оцінка прогноза:")
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

# Прогнозування окремого входження за допомогою натренованої моделі
newdataset = np.array([[5, 2.9, 1, 0.2]])

svm_model = [model for name, model in models if name == 'SVM'][0]
svm_model.fit(X_train, Y_train)
prediction = svm_model.predict(newdataset)
print("Прогноз: " + str(prediction[0]))

```

Виведення в консоль:

```

(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)
Оцінка прогноза:
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
[[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         30
 weighted avg                   0.97         30

Прогноз: Iris-setosa

```

Отже, якість класифікації 0.97, а спрогнозовано було, що описана нами квітка входить до виду “Iris-setosa”

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

Програмний код:

```

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
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
from sklearn import preprocessing
import numpy as np

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.strip().split(',')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data[:-1])
            y.append('<=50K')
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data[:-1])
            y.append('>50K')
            count_class2 += 1

# Перетворення на масив numpy
X = np.array(X)
y = np.array(y)

# Перетворення рядкових даних на числові
label_encoders = []
X_encoded = np.empty(X.shape)

for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i].astype(int)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders.append(le)

# Поділ на навчальні і тестові дані
X_train, X_validation, y_train, y_validation = train_test_split(X_encoded, y,
test_size=0.2, random_state=5)

# Завантажуємо алгоритми моделі
models = []
#models.append(('LR', LogisticRegression(solver='liblinear',multi_class='ovr')))
models.append(('LR', OneVsRestClassifier(LogisticRegression()))) #LR multi_class
був деприкейтед говорить
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))

```

```

models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(kernel='poly', degree=2)))
results = []
names = []

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

```

Результат:

```

LR: 0.784699 (0.004054)
LDA: 0.811637 (0.005701)
KNN: 0.767748 (0.003026)
CART: 0.806706 (0.007076)
NB: 0.789133 (0.006934)
SVM: 0.781176 (0.003242)

```

## Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Програмний код:

```

# =====
# Приклад класифікатора Ridge
# =====

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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 io import BytesIO

# Завантаження даних Iris
iris = load_iris()
X, y = iris.data, iris.target

# Розділення на тренувальну і тестову вибірки
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3,
random_state=0)

# Створення та навчання класифікатора Ridge
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)

# Прогнозування
ypred = clf.predict(Xtest)

```

```

# Оцінка якості моделі
print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))
print('Precision:', np.round(metrics.precision_score(ytest, ypred,
average='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))

# Звіт про класифікацію
print('\t\tClassification Report:\n', metrics.classification_report(ytest,
ypred))

# Побудова матриці плутанини
mat = metrics.confusion_matrix(ytest, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, cmap="Blues")
plt.xlabel('True label')
plt.ylabel('Predicted label')

# Збереження зображення у форматі JPG
plt.savefig("Confusion.jpg")

# Збереження зображення у форматі SVG у віртуальний файл
f = BytesIO()
plt.savefig(f, format="svg")
plt.show()

```

Результат:

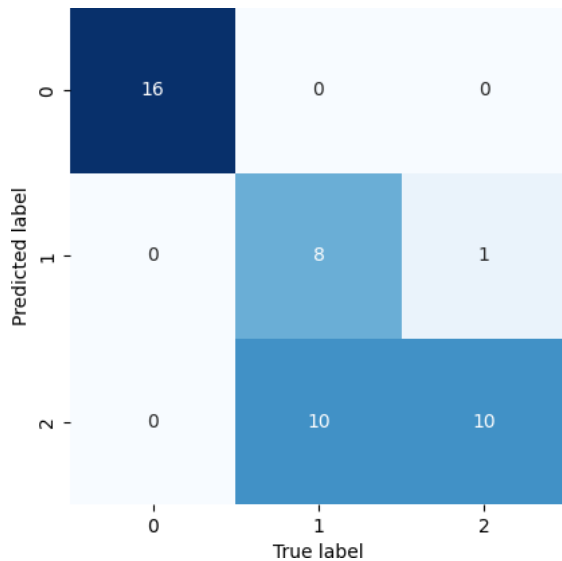
```

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

Матриця плутанини:



Тут використовується класифікатор Ridge (RidgeClassifier) із параметрами  $\text{tol}=1\text{e-}2$  і  $\text{solver}=\text{"sag"}$ . Параметр  $\text{tol}=1\text{e-}2$  визначає допустиму похибку для критерію зупинки оптимізації, що означає, що алгоритм припинить навчання, коли зміни у втраті будуть меншими за 0.01. Параметр  $\text{solver}=\text{"sag"}$  вказує, що використовується стохастичний градієнтний метод SAG (Stochastic Average Gradient), який добре підходить для великих наборів даних.

Для оцінки якості моделі обчислюються наступні метрики. Accuracy (точність) відображає відсоток правильно класифікованих об'єктів серед усіх тестових зразків. Precision (прецизійність) оцінює частку правильно передбачених позитивних класів від усіх, передбачених як позитивні. Recall (повнота) показує, яку частку об'єктів певного класу класифікатор правильно ідентифікував. F1 Score є середнім гармонічним між precision і recall, що дає баланс між ними. Cohen Kappa Score (коефіцієнт Каппа Коена) вимірює узгодженість між передбаченнями моделі та реальними мітками, враховуючи випадкову згоду. Matthews Correlation Coefficient (коефіцієнт кореляції Метьюза) оцінює якість класифікації, беручи до уваги всі чотири значення матриці плутанини (TP, TN, FP, FN), і є особливо корисним для задач із незбалансованими класами.

Отримані результати показують, що класифікатор має досить високу точність, оскільки значення метрик близькі до 1. Побудована матриця плутанини (Confusion.jpg) відображає, як модель помиляється у визначенні класів. Кожна клітинка цієї матриці містить кількість передбачень для відповідної комбінації "справжнього класу" (по горизонталі) та "передбаченого класу" (по вертикалі). Значення на головній діагоналі вказують на правильно класифіковані об'єкти, а поза нею — помилки. Світліші клітинки візуально показують більші значення, тобто ті категорії, які були найкраще передбачені.

Коефіцієнт Коена Каппа коригує показник точності, враховуючи випадкові збіги, і має значення від -1 до 1, де 1 означає повну узгодженість, 0 — узгодженість випадкову, а від'ємні значення — гірше за випадкове вгадування. Тут він вимірює наскільки узгоджені результати моделі з реальними мітками класів, коригуючи на ймовірність випадкової угоди. Коефіцієнт кореляції Метьюза також оцінює якість класифікації та є більш надійною метрикою для незбалансованих наборів даних, оскільки враховує всі елементи матриці плутанини. Він варіюється від -1 до 1, де 1 означає ідеальну класифікацію, 0 — випадкове вгадування, а -1 — повне невідповідність передбачених значень реальним. У цьому випадку ці коефіцієнти підтверджують високу якість класифікації.

**Висновок.** У цій лабораторній роботі було проведено класифікацію даних за допомогою машинного навчання. Основну увагу приділено вибору відповідної моделі, її налаштуванню та оцінці якості класифікації. Було виконано попередню обробку даних, використано різні алгоритми класифікації та проведено їх порівняльний аналіз. Отримані результати свідчать про ефективність обраних методів та важливість коректного налаштування параметрів моделі для досягнення високої точності прогнозування.

